2015–1734, –1736, –1737, –1738, –1739, –1740, –1741, –1742,
–1816, –1817, –1818, –1819

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

MICROSOFT CORPORATION,

Petitioner–Appellant,

v.

ENFISH, LLC,

Respondent–Appellee

Appeals from the United States Patent and Trademark Office, Patent Trial and Appeal Board in case nos. IPR2013–00559, IPR2013–00560, IPR2013–00561, IPR2013–00562, IPR2013–00563

APPELLANT'S OPENING BRIEF

Chad S. Campbell
Dan L. Bagatell
Theodore H. Wimsatt
Jonathan D. Allred
PERKINS COIE LLP
2901 N. Central Avenue, Suite 2000
Phoenix, Arizona 85012–2788
Phone: (602) 351–8383
E-mail: CSCampbell@perkinscoie.com

Amy E. Simpson
PERKINS COIE LLP
11988 El Camino Real, Suite 300
San Diego, California 92130-3334
Phone: (858) 720–5702
E-mail: ASimpson@perkinscoie.com

Attorneys for Microsoft Corporation

November 20, 2015

CERTIFICATE OF INTEREST

Counsel for petitioner-appellant Microsoft Corporation certifies the following:

The full name of every party represented by me is:

Microsoft Corporation.

The names of the real parties in interest (if the parties named in the caption are not the real party in interest) represented by me are:

n/a

The names of all parent corporations and any publicly held companies that own 10% or more of the stock of the party represented by me are:

none

The names of all law firms and lawyers that appeared for the parties now represented by me at the Patent and Trademark Office or that are expected to appear in this Court are:

PERKINS COIE LLP

Jonathan D. Allred	Dan L. Bagatell	Chad S. Campbell
Chun M. Ng	Amy E. Simpson	Theodore H. Wimsatt

Dated: November 20, 2015

/s/Chad S. Campbell
Chad S. Campbell

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases **Pages**

Statutes and Regulations **Pages**

TABLE OF ABBREVIATIONS AND CONVENTIONS

A___	joint appendix page ___
Anderson	U.S. Patent No. 5,463,724—secondary reference for self-referential table claims
Board or PTAB	Patent Trial and Appeal Board of the United States Patent and Trademark Office
Chang	Chang et al., EP Pub. No. 0 336 580 A2 (published Oct. 11, 1989)—primary reference for self-referential table claims
Enfish	respondent-appellee Enfish, LLC
Goldberg	U.S. Patent No. 5,201,046—secondary reference for self-referential table claims
Horn	U.S. Patent No. 5,226,158—secondary reference for self-referential table claims
IPR	*inter partes* review
Microsoft	petitioner–appellant Microsoft Corporation
OID	object identification number
Salton	Gerard Salton & Michael J. McGill, *Introduction to Modern Information Retrieval* (1983)—secondary reference for indexing claims
Visual Basic manual or VB3	Microsoft Visual Basic Programming System for Windows Version 3.0 (1993)—primary reference for indexing claims
’604 patent	U.S. Patent No. 6,151,604
’775 patent	U.S. Patent No. 6,163,775
’###(*xx*:*yy*-*zz*)	column *xx*, lines *yy* to *zz* of the ’### patent

RELATED CASES

No other appeals from these five *inter partes* reviews (IPRs) before the Patent Trial and Appeal Board (PTAB) have previously been before this or any other appellate court. The two patents at issue here were also at issue in a civil action that Enfish filed against Microsoft and others in the United States District Court for the Central District of California, and the judgment in that case is now on appeal to this Court in appeal No. 2015–1244. An affirmance of the district court's judgment would moot some but not all of the issues in these IPR cases.

Microsoft and its counsel are not aware of any other cases pending in this or any other court that will directly affect or be directly affected by the decision in this case.

INTRODUCTION

This appeal involves five IPR petitions challenging the validity of claims of U.S. Patent Nos. 6,151,604 and 6,163,775. The Patent Trial and Appeal Board correctly found that many of the claims were indefinite and that all remaining independent claims were invalid over prior art. The Board also ruled, however, that Microsoft failed to prove the invalidity of certain dependent claims. This appeal challenges some (not all) of the latter rulings. As shown below, the appealed determinations should be reversed or at least vacated and remanded for reconsideration under the correct legal standards. In particular:

- Although the Board correctly found that a published prior-art application by Chang anticipated claim 31 of each patent, the Board erred in finding that Chang did not anticipate dependent claim 32 of each patent. Claims 32 require a table column that is defined to enable determination of object identification numbers ("OIDs") from text entry. Chang disclosed tables with columns to be used for determining such OIDs upon keyboard entry of queries. Chang also disclosed columns holding index information for text searching. In discounting those teachings, the Board adopted two requirements: (i) enabling OID determination from text *searching* (rather than text entry), and (ii) searching based on the column information alone. But claims 32 impose neither requirement.

- The Board's non-invalidity determinations regarding claims 36 and 37 of each patent and claim 42 of the '775 patent should be vacated because the Board did not apply the broadest reasonable interpretation of the claims, as its rules required. Claims 36 require a row to include information defining the type of "a different logical row," but nothing in the patents requires that "different logical row" to be within the same table rather than a related table as in Chang. Claim 37 of both patents and claim 42 of the '775 patent require "pointers," but nothing in the patents requires pointing to particular *addresses* in memory, as opposed to other locations within the database, as shown in a prior-art reference.

- Although the Board correctly found independent claim 42 of the '604 patent obvious over prior art, it erred in finding dependent claim 43 not invalid. The Board misunderstood Microsoft to be relying on a combination of two tables, when in fact it was relying on one. This error appears to have resulted from misreading claim 43 to require pointers from rows to other *rows*, when the additional limitation of claim 43 actually requires pointers to other *columns*.

- Although the Board correctly found independent claim 54 of both patents invalid for anticipation, it erred in upholding the validity of dependent claims 55, 56, and 60. Here the Board's error was in finding no motivation to combine the high-level description of indexing in the product manual that anticipated claims 54 with the low-level implementation details supplied by a standard

textbook discussion of indexing. An ordinary artisan seeking more details about how to index naturally would have turned to a standard textbook reference.

JURISDICTIONAL STATEMENT

The Board had jurisdiction under 35 U.S.C. § 318(a) and issued final written decisions. A1; A53; A88; A135; A164. Microsoft timely petitioned for rehearing in two cases, A1347; A1366, which suspended the deadline to appeal in those cases under 37 C.F.R. § 90.3(b)(1). After the Board denied rehearing, A1358; A1377, Microsoft filed timely notices of appeal in those cases, 15-1816 Dkt. 1-2; 15-1818 Dkt. 1-2. In the other three cases, Microsoft filed timely notices of appeal directly from the final written decisions. 15-1737 Dkt. 1-2; 15-1739 Dkt. 1-2; 15-1741 Dkt. 1-2. This Court has appellate jurisdiction under 35 U.S.C. §§ 141-144 and 319.

STATEMENT OF ISSUES

1. Chang taught the core claimed concept of self-referential tables that include definitions of columns in rows of the table. The Board concluded that Chang's tables anticipated claim 31 of both patents, but not dependent claims 32, which further require column information "for enabling determination of OIDs from text entry." The Board reasoned that Chang's tables did not clearly show how to perform text searching with the column information "alone," but neither

text searching nor searching using column information alone is a claim requirement. Should the Board's non-anticipation determination be reversed?

2. Claim 36 of each patent requires that "at least one of said plurality of logical rows includes information defining the type of a different logical row." The claims do not require a row to define the type of "one of said plurality of logical rows," only the type of "a different logical row." Nevertheless, the Board imported a limitation requiring the row to define the type of a row in the same table rather than a related table and upheld the claims' validity on that basis. Did the Board fail to apply the broadest reasonable interpretation, requiring remand for reconsideration under the proper, broader construction?

3. Claim 37 of both patents and claim 42 of the '775 patent require "pointers" indicating which columns within the same record contain defined values. The OIDs described in the patents as "pointers" are not addresses of locations in memory. Nevertheless, the Board upheld these claims based on a narrow construction requiring "pointers" to refer to an "address where another object resides." Did the Board fail to apply the broadest reasonable interpretation, requiring remand for reconsideration under the proper, broader construction?

4. Claim 43 of the '604 patent requires a column to define cells that include pointers to other columns within the same record that contain defined values. In upholding this claim, the Board assumed that Microsoft was relying on

the combination of two tables to satisfy this limitation. In fact, Microsoft relied on a single table. Should this Court vacate and remand for the Board to consider Microsoft's actual invalidity contention?

5. Dependent claims 55, 56, and 60 of each patent specify more detail about the indexing required by independent claim 54. In particular, claims 55 and 56 require creating an index record for a keyword having a pointer to a cell in the table that contains that keyword, while claims 60 require indexing to include indexing of external documents. Microsoft's lead reference for claim 54 was a product manual that described using keyword indexes but did not delve into all of the details about how to create such indexes. Microsoft's invalidity contentions for claims 55, 56, and 60 supplemented the product manual with a well-known textbook describing widely used indexing principles and techniques. The Board upheld the claims on grounds that there was no motivation to combine the details from the textbook with the higher-level description of indexing in the product manual. Did the Board err in finding no motivation to combine the references?

STATEMENT OF THE CASE

I. Enfish and the patents on appeal

Enfish's '604 and '775 patents are continuations from a common parent and share nearly identical written descriptions and figures. Because the claims in both patents are nearly identical, this brief refers primarily to the '604 patent.

The patents are directed to databases that organize data into logical tables consisting of rows, which serve as "records" about items in the database, and columns, which serve as "attributes" of the records. The intersection of a row and column forms a "cell." The rows and columns have identifiers called "object identification numbers" or "OIDs." The data in a table may be indexed to aid searching.

Databasing was highly developed by the time of the alleged invention. Indeed, both storing data in tables with intersecting rows and columns having unique identifiers and indexing the data in a table were concededly conventional concepts by the 1995 filing date of the parent application. '604(1:47-56); A7661-62 (explaining ubiquity of indexing). The patents instead suggest two supposed improvements over previously known table techniques.

First, the patents describe including in the same table both the information that the table organizes and definitions of the table's columns (sometimes called the table's "schema"). The patents refer to such tables as "self-referential." Figure 3 illustrates the concept. The added red highlighting shows where the table includes a row (136) that stores the column name for one of the columns:

FIG. 3

120 122 130 124 134 126 132 100

108	OBJECT ID	TYPE [# 101]	[#1012] LABEL	ADDRESS [#1013]	EMPLOYED BY [#1019]	TITLE [#1033]	AUTHOR [#1032]
110	#1100	#1020 [COMPANY]	DEXIS	117 EAST COLORADO		N/A	N/A
138	#1101	#1010 [PERSON]	SCOTT WLASCHIN		#1100 [DEXIS]	N/A	N/A
	#1118	#1030 [BOOK]					#1122
	#1122	#1050 [MEMO]					#1122
	#1127	#1060 [DOCUMENT]		C:\WORD\PROJ.DOC		PROJECT PLAN	#1101
136	#1019	# 210 [FIELD]	EMPLOYED BY				
135	# 210	# 111 [TYPE]	COLUMN				
140	# 111	# 111 [TYPE]	TYPE				

133

Second, the patents say that the table may include index records. Such records are rows that store, for example, a keyword extracted from another record in the database along with the location where the keyword can be found. The patents teach to put pointers to the index records and/or keywords in cells of the table. '604(Fig. 14, 12:16-19, 14:14-17).

II. The claims on appeal

This appeal involves 14 dependent claims, eight directed to self-referential tables and six directed to indexing.

A. The self-referential table claims

Claims 32, 36, and 37 of each patent depend from claim 31, which is not at issue here because the Board found it invalid for anticipation. Claim 31 of the ’604 patent recites a table with multiple rows and intersecting columns that each have an OID. The final paragraph of claim 31 adds the self-referencing concept by requiring at least one row to have an OID equal to the OID of a column and at least one row to hold column defining information for the columns:

> 31. A method for storing and retrieving data in a computer memory, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;
>
> a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein
>
> at least one of said logical rows has an OID equal to the OID to a corresponding one of said logical columns, and at least one of said logical rows includes logical column information defining each of said logical columns.

’775 claim 31 is essentially similar but substitutes the synonym “record” for “row” and the synonym “attribute set” for “column.”

Dependent claims 32, 36, and 37 of each patent add further requirements such as a column defined to hold information enabling determination of OIDs in the database through text entry, using at least one row to define the type of a different row, and including in rows and cells pointers to other rows or columns. The ’604 claims recite:

> 32. The method of claim 31 wherein said logical column information defines one of said logical columns to contain information for enabling determination of OIDs from text entry.
>
> …
>
> 36. The method of claim 31 wherein:
>
> at least one of said plurality of logical rows includes information defining the type of a different logical row; and
>
> at least one of said plurality of logical rows includes a logical cell that contains a pointer to said logical row including logical row type information.
>
> 37. The method of claim 31 wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

Again, ’775 claims 32, 36, and 37 are essentially similar.

Claim 42 is the other relevant independent claim of the ’604 patent involving self-referential tables. It is not at issue here because the Board found it

obvious. Claim 42 resembles claim 31 but also requires a row with a cell containing a pointer to a different row:

> 42. A method for storing and retrieving data in a computer memory, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;
>
> a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein
>
> at least one of said logical rows contains a logical cell that contains a pointer to a different logical row and at least one of said logical rows includes logical column information defining each of said logical column; and
>
> searching said table for said pointer.

’775 claim 41 (*sic*) is essentially similar.

Dependent claim 43 of the ’604 patent, which is at issue here, adds a requirement that cells include pointers to other columns:

> 43. The method of claim 42 wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

’775 claim 42 (*sic*) is essentially similar.

B. The indexing claims

Claims 55, 56, and 60 of each patent depend from claim 54, which is not at issue here because the Board found it anticipated. Claim 54 of the ’604 patent is similar to claim 31 except that instead of rows defining columns, it requires indexing data in the table and including at least one cell with a pointer to an index record:

> 54. A method for storing and retrieving data in a computer memory, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;
>
> a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein
>
> at least one of said logical cells includes a pointer to an index record; and
>
> indexing data stored in said table.

’775 claim 54 is essentially similar.

Dependent claims 55, 56, and 60, which are at issue here, add requirements of indexing keywords in the table, locating index records based on a user's query, or indexing external documents. The '604 claims recite:

> 55. The method of claim 54 wherein said step of indexing data further comprises the steps of:
>
> searching said table for a key word; and
>
> creating an index record for said key word, said index record having one or more pointers to a logical cell in said table that contains said key word.
>
> 56. The method of claim 55 further comprising the steps of:
>
> locating said index record according to the query of a user;
>
> retrieving at least one logical cell in said table pointed to by said located index record.
>
> …
>
> 60. The method of claim 54 wherein said step of indexing data further comprises the step of:
>
> indexing external documents.

The '775 counterparts are essentially similar.

III. The prior-art patents and publications

A. Chang taught self-referential tables and columns that enabled determining OIDs from text entry

The principal prior-art reference showing self-referential tables was an IBM patent application by Chang that was published in 1989. The Board found that

Chang anticipated claim 31 of each patent and rendered other claims obvious when combined with supplemental references. A50; A132.

Chang disclosed two self-referential tables called "catalogs"—specifically, SYS.TABLES in Figure 2 and SYS.COLUMNS in Figure 3. A3763-79.[1] Those tables were part of a relational database that included various tables that each held rows/records of related items. A3773.

The SYS.COLUMNS table included a row/record for each column of every table throughout the system. Each row held the definition (schema) of its corresponding column. Because SYS.COLUMNS included a column-defining row for every column of every table, it also included a row to define each of its own columns, rendering it self-referential. Enfish itself illustrated that aspect of SYS.COLUMNS by augmenting Chang's Figure 3 in this way (with rows 9-12 holding the column definitions for the first four columns of SYS.COLUMNS):

[1] Chang referred to SYS.TABLES and SYS.COLUMNS as "catalogs" but observed that system catalogs are tables. A3767(8:45-46).

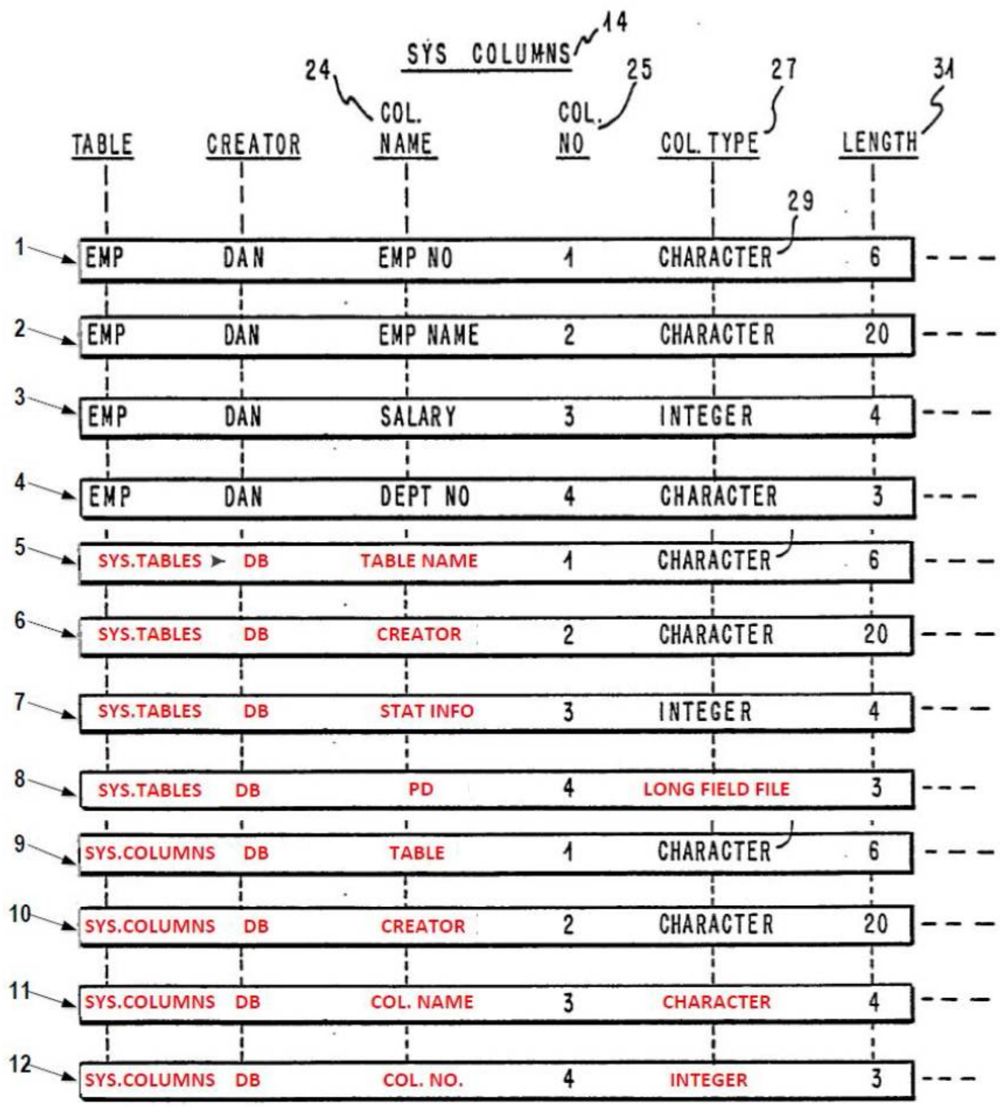


FIG. 3 (Annotated)

A981.

For its part, SYS.TABLES held a row/record for each table in the database. Each row held the definition (schema) for its corresponding table. Because SYS.TABLES included a row for each table, it also included a row defining itself, rendering the table self-referential. In particular, the table definition of a row in

SYS.TABLES appeared in a "packed descriptor" or "PD" column, as illustrated in Chang's Figure 2. A3773. Although Figure 2 literally showed only the row corresponding to a table of employees in the database, Chang made clear that SYS.TABLES contained a row with a packed descriptor for each table in the database. For each table, the packed descriptor included the definition of every column in the table as well as the schema for any set of index records that went with the table. SYS.TABLES was self-referential because it included a row with a packed descriptor that defined the SYS.TABLES table itself and that packed descriptor included a definition of each column in SYS.TABLES. A229-30; A363-66; A992; A3766-67(5:54-7:19).

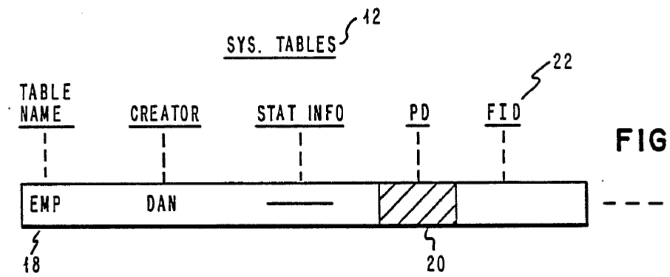


FIG. 2

The rows and columns of both SYS.COLUMNS and SYS.TABLES contained multiple bit arrays that identified them. The rows were each associated with a "record i.d." A3767(7:51-8:3); A225, A232; A357, A361-62. In addition, the combination of the table name and column number values identified the rows of SYS.COLUMNS, and the combination of the table name values (or table name + creator name) identified the rows of SYS.TABLES. A225, A232; A357, A361-62. The columns in each table were individually identified by the column number and a column name. A226; A358-59; A1574-75 ¶¶ 167-168.

Like relational databases generally, Chang's system was constructed to permit users to access rows by entering queries on a keyboard. Chang's Figure 11 illustrated such an arrangement, with a keyboard and monitor connected to a processor and storage device. A3779. Using any conventional query language (e.g., SQL), a

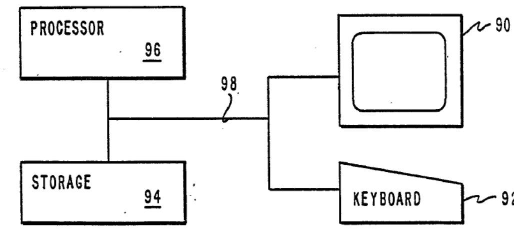


FIG. 11

user would type in the text of a query, and the system would return the record or records that matched the query. A3764(1:26-43), A3770(13:45-14:10); A1571 ¶ 158; A2696.

The retrieval process required the system to validate a query against the schema of the queried table, which required accessing the column definitions for the table. Those definition checks could be done one at a time through accesses to the relevant records of SYS.COLUMNS. A3764(2:4-21), A3767(8:4-20). Alternatively, the definitions could be obtained with a single access to the applicable packed descriptor in SYS.TABLES. A3768(9:23-33). Either way, accessing the definitions yielded the column identifiers, including column names and column numbers.

B. Goldberg taught rows that define other row types in a table

The Goldberg patent was a secondary prior-art reference for the self-referential claims that described an improved relational database for storing data

with a hierarchical relationship between items, such as an auto-parts database. A3805(3:47-62, 4:45-63). Goldberg categorized records by including primary key pointers in a cell within the record. A3801-02(Figs. 4, 6), A3810(13:2-4).

C. Anderson taught using cells with pointers to other columns in a given row

The Anderson patent was a secondary reference for the self-referential claims that described aspects of the Quattro Pro spreadsheet program, a type of tabular database. A1622 ¶ 321. Among other things, Anderson disclosed storing within one cell locations of other cells with pertinent data. A3831; A1622 ¶ 322; A2751-52.

D. Horn also taught pointers to other columns

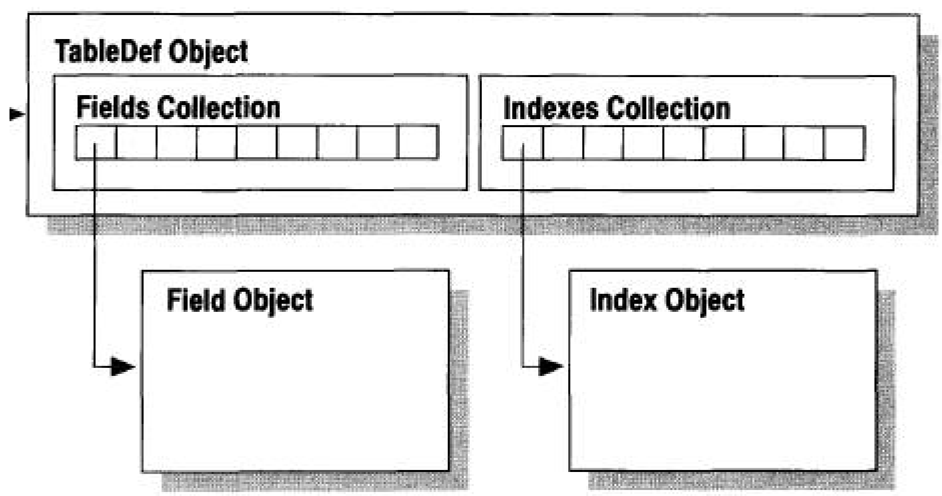
The Horn patent was a secondary reference for the self-referential claims that described an improved relational database that could synchronize data in two tables. A3780(Abstract). Among other things, Horn explained that some tables have relationships to other tables that require synchronization, A3791(1:42-61), and depicted a well-known technique of pointers from one row to another within the same table, A3782; A1610 ¶ 270; A2733-34.

E. The Visual Basic manual taught indexing data in a table and pointers to another row

For the indexing claims, Microsoft relied primarily on a product manual that described Microsoft's Visual Basic 3 database product. The manual disclosed a

programming framework that included Table Objects and indexes that programmers could use to manage data from a variety of sources. Each Table Object comprised records (rows) and fields (columns). For identification purposes, unique values were assigned to both records and fields. For example, each record had a "primary key" for determining "if the record [wa]s unique within the table," A7566, while each column was assigned an ordinal value ("0" to the first field, "1" to the second, etc.), as well as a unique name to uniquely identify that field, A7275; A7240; A7607.

To organize data for efficient retrieval, the Visual Basic manual taught to use an Index. A7126-27; A6612. In particular, it taught to store pointers to that Index within the Table Object, as shown below:



A6614.

F. Salton taught several well-known approaches to indexing data in a table, including the claimed keyword index

The 1983 textbook *Introduction to Modern Information Retrieval* by Gerald Salton and Michael J. McGill ("Salton") was well known in the field of information retrieval, *see* A7897, and specifically taught several ways of indexing information. In its first chapter, entitled "Information Retrieval: An Introduction," Salton introduced several "simple file structures" for efficient information retrieval, including an "inverted file" (index) including keywords and the locations where they could be found—similar to a topical index at the back of a book containing an alphabetized list of topics and the page numbers where those topics are found. A7649, A7653-57. Such indexes were common even in 1983, Salton noted, as they were "used in almost every commercially available information retrieval system." A7655. Salton also included details about various implementations of inverted indexes, A7661-88, as well as specific indexing strategies such as full-text indexing and automatic indexing, A7689-7754.

IV. The district court litigation

Although the original version of Enfish failed in the marketplace and went bankrupt, its founder salvaged the patents in bankruptcy and formed the current incarnation of Enfish as a patent-assertion entity. In 2012, Enfish sued Microsoft and several others on the '604 and '775 patents. Enfish alleged infringement by Microsoft's ADO.NET software, a library of routines useful for writing programs

to run on Windows® operating systems. ADO.NET can help translate data from various formats into a standard format, and Enfish contended that features tied to multiple DataTable objects in ADO.NET satisfied the elements of eleven claims: ’604 claims 1, 2, 16, 17, 31, 32, 46, and 47 and ’775 claims 31, 32, and 47. A5384-5430.

Following claim construction and four motions for summary judgment, the district court ruled that three system claims (’604 claims 1, 2, and 16) were indefinite for improperly reciting a single means-plus-function limitation; seven method claims (’604 claims 31, 21, 46, and 47 and ’775 claims 31, 32, and 47) were anticipated by the prior-art software product Excel 5.0; one claim (’604 claim 17) was not infringed; and all eleven asserted claims were invalid for claiming unpatentable subject matter. Enfish has appealed to this Court from the resulting judgment, but only as to ’604 claims 17, 31, and 32 and ’775 claims 31 and 32. That appeal (No. 2015-1244) is fully briefed. Because Enfish has not appealed the rulings declaring ’604 claims 1, 2, 16, 46, and 47 and ’775 claim 47 invalid, there is no live dispute over those claims.

V. The IPRs and the PTAB’s final written decisions

After Enfish filed suit but before the district court issued its summary judgment rulings, Microsoft petitioned for *inter partes* review. Not knowing which claims Enfish would assert in the litigation, Microsoft petitioned for review of all

120 claims in the two patents. The Board instituted review of 114 of the claims and ultimately issued a mixed set of rulings that terminated review as to 54 claims due to indefiniteness, canceled 18 claims for invalidity over prior art, and upheld the patentability of 36 other claims. A50-51; A86; A132; A162; A191.

A. The Board's claim constructions

The final written decisions expressly construed the term "object identification number" or "OID," which appears in each claim. In its institution decisions, the Board provisionally construed "identification number" to mean "an array of bits that define" and indicated that an express construction of "object" was unnecessary. A816; A849; A877; A912; A938. Enfish agreed with that "in principle" but urged additional limitations requiring OIDs to be universally unique throughout a database, immutable, and system-generated. A969-70. The Board declined those additions because the preferred embodiment included two different objects that had the same OID, the intrinsic record mentioned no immutability requirement, and a system-generation requirement would have improperly imported a feature of a disclosed embodiment into the claims. A17-19.

Although the Board did not expressly construe other terms at issue, its decisions on the dependent claims at issue here effectively adopted and applied narrowing claim interpretations that Microsoft disputes. Those constructions are discussed below.

B. The Board's rulings on the self-referential method claims

Claims 31. The Board concluded that Chang anticipated independent method claim 31 of both patents. Although Microsoft argued that both SYS.COLUMNS and SYS.TABLES anticipated, the Board's final written decisions addressed only SYS.COLUMNS. The Board rejected Enfish's argument that SYS.COLUMNS lacked OIDs for rows, which depended on Enfish's overly narrow interpretation of "OID." The Board also rejected Enfish's argument that SYS.COLUMNS lacked any row having an OID of a corresponding column because SYS.COLUMNS included rows that stored the definitions and identifiers for each of the columns in the SYS.COLUMNS table. A25-26; A109-11.

Claims 32. Microsoft contended that Chang (in particular SYS.COLUMNS and SYS.TABLES) also anticipated dependent claim 32 of both patents, but the Board disagreed. A26-27; A111-13. Because the Board appeared to have overlooked one of Microsoft's arguments, Microsoft sought rehearing on claim 32 of both patents, without success. A1359; A1378. In both rulings, the Board reasoned that Microsoft had not shown how a defined column in SYS.COLUMNS or SYS.TABLES *alone* contained information that could be searched to determine an OID. A1359-60; A1378-79.

Claims 36 & 37. The Board upheld dependent claims 36 and 37 of both patents over the combinations of Chang + Goldberg and Chang + Anderson,

respectively. A42-46; A122-26. The Board read claims 36 to require that one row define the type of a different row in the same table and faulted Microsoft's position for relying on two related but distinct tables in Chang. A43; A123-24. Regarding claims 37, the Board limited the recited "pointers" to variables that store a database *address* where another object resides and reasoned that the cell pointers in Anderson on which Microsoft relied did not store memory addresses. A45-46; A125-26.

'604 Claims 42 and 43. The Board concluded that claim 42 of the '604 patent was obvious over Chang combined with the well-understood concept of using row-to-row pointers to create relationships within a table as exemplified by the employee table of Horn. A28-31. The Board rejected Enfish's argument that the employee numbers of Horn did not function as addresses pointing to the location of their associated employees. A29-30. The Board upheld dependent claim 43, however, concluding that Microsoft's reliance on the packed descriptor column of SYS.TABLES for the "pointer" limitation of that claim relied on a combination of two tables rather than one. A31.

'775 Claims 41 and 42. Claim 41 of the '775 patent is essentially like '775 claim 31 but with an additional limitation requiring "searching said table for said pointer." A2708 ¶ 204. The Board adopted its reasoning for claim 31, noted that Enfish made no argument regarding the additional limitation in claim 41, and found that Chang disclosed searching an index to return a row or rows from which

the OID acting as a pointer could be determined. A111. The Board upheld dependent claim 42, however, based on its interpretation that the claimed "pointers" had to include "addresses." A126.

C. The Board's rulings on indexing method claims

Claims 54. Microsoft argued that tables detailed in the Visual Basic user manual anticipated claim 54 of both patents. A69-70; A179-80. The Board agreed, noting that tables in the manual had multiple rows and multiple columns, each with an object identifier, along with an index of data stored in the table and a cell holding a pointer to an index record. A69-71; A179-81. The Board rejected Enfish's challenges to the presence of OIDs in those tables because Enfish's challenges were based entirely on its unduly narrow definition of "OID." A70-71; A180-81.

Claims 55, 56 & 60. For claim 55 of both patents, Microsoft argued that the Visual Basic manual combined with part of the Salton textbook made the claims obvious. A318-23; A504-08, A510-13. The Board concluded, however, that Microsoft had not shown an adequate motivation to combine Salton's recitation of the already-widespread practice of keyword indexing in databases with the indexed tables of Visual Basic. A76-78; A183-85. The Board applied the same reasoning in upholding claims 56 and 60 of both patents over the combination of Visual Basic and Salton. *Id.*

SUMMARY OF ARGUMENT

1. The Board's determination that Chang's tables did not anticipate claim 32 of each patent should be reversed because the decision depends on requirements that the claims do not have. Chang's tables defined columns for access to and determination of OIDs upon text entry of user queries. The Board discounted those teachings on grounds that they did not clearly disclose how text searching was involved in the process. But claims 32 do not recite text *searching*, only text *entry*. Indeed, the claim language was broadened during prosecution by eliminating the term "searching," and some embodiments disclosed in the written description do not require searching. Chang also taught columns that hold certain indexing information, but the Board set that teaching aside because the columns supposedly were not sufficient to complete the searching task by themselves. Here too, the Board erred because claims 32 contain no such requirement.

2. The Board's non-invalidity determination regarding claim 36 of each patent should be vacated because the Board failed to apply the broadest reasonable interpretation. Claims 36 require "at least one of said plurality of logical rows includes information defining the type of a different logical row." In requiring the "different logical row" to be within the same table, the Board improperly rewrote "a different logical row" as "a different one of said plurality of logical rows" in the table even though neither the claim language nor the specification so dictates and

the prosecution history is contrary. Under the broadest reasonable interpretation, "a different logical row" may be a row in another, related table.

3. The Board's non-invalidity determinations regarding claim 37 of each patent and claim 42 of the '775 patent should also be vacated because the Board again did not apply the broadest reasonable interpretation. These claims recite "pointers" to other locations in the database. The Board required "pointers" to refer to addresses where other objects reside, but the claims do not mention addresses and the specification describes OIDs as "pointers" to other records even though those OIDs are not addresses of locations in memory. Under the broadest reasonable interpretation, a "pointer" need only refer to (and thus provide a shortcut to) another record or cell.

4. The Board's non-invalidity determination regarding claim 43 of the '604 patent should be vacated because the Board misunderstood Microsoft's invalidity argument. The Board thought Microsoft was relying on a combination of two tables from Chang to satisfy the limitation requiring pointers to other logical columns containing defined values, but Microsoft actually relied on a single table (SYS.TABLES) to meet this limitation. The Board reasoned that the packed descriptor of SYS.TABLES identified rows within SYS.COLUMNS, but whether that is so is irrelevant because claim 43 does not require pointers to rows. Claim

43 requires pointers to other logical *columns* within a record, and Microsoft showed that the column numbers of SYS.TABLES identified locations of columns.

5. The Board's non-invalidity ruling regarding dependent claims 55, 56, and 60 of each patent should be reversed because the Board erred in finding no motivation to combine the two cited references. These claims provide more detail about the indexing required by independent claims 54. The Visual Basic product manual described indexing data in a table as required by claims 54, but it did not describe some of the implementation details contained in claims 55, 56, and 60. A skilled artisan needing more information about how to implement indexing naturally would have turned to a standard textbook discussion of how to index. The Salton textbook on which Microsoft relied provided all the necessary supplemental information. In finding no motivation to combine the Visual Basic manual with the Salton textbook, the Board applied rigid rules contrary to the flexibility required by *KSR* and later cases from this Court. Combining a high-level description in one reference with more details from a standard reference was a matter of common sense for even an inexperienced software engineer.

ARGUMENT

By rule, the Board must give claims their "broadest reasonable construction" during *inter partes* reviews of unexpired patents (which these were at the time of the final written decisions). 37 C.F.R. § 42.100(b). This Court upheld that rule in

In re Cuozzo Speed Technologies, LLC, 793 F.3d 1268, 1279 (Fed. Cir. 2015). On appeal, this Court reviews the Board's ultimate claim constructions *de novo*, applying the same broadest-reasonable-interpretation standard. *Id.* at 1280. To the extent the Board made factual findings based on extrinsic evidence in construing the claims, this Court reviews those findings for substantial evidence. *Id.*

This Court reviews the Board's factual findings regarding anticipation for substantial evidence. *Power Integrations, Inc. v. Lee*, 797 F.3d 1318, 1323 (Fed. Cir. 2015). This Court reviews the Board's ultimate legal determinations of non-obviousness *de novo* but reviews any underlying factual findings for substantial evidence. *Randall Mfg. v. Rea*, 733 F.3d 1355, 1362 (Fed. Cir. 2013).

Under these standards, this Court should hold that the Board erred in determining that the appealed claims are not invalid and reverse or remand for reconsideration of validity of those claims under the correct framework.

I. Chang anticipated claim 32 of each patent under a proper construction

Although the Board found claim 31 of each patent anticipated by Chang, it ruled that Microsoft had not shown anticipation of dependent claims 32. That determination should be reversed because the same features in Chang that met the limitations of claims 31 also satisfied the additional requirement of claims 32. The Board's contrary conclusion was mistaken because it improperly imported requirements into the claims.

A. The Board correctly concluded that Chang anticipated claim 31 of each patent

Enfish did not dispute that Chang's SYS.COLUMNS table was a logical table with multiple logical rows that intersected multiple logical columns to form logical cells. Enfish also did not dispute that each column in the SYS.COLUMNS table had a column number and a column name and that its rows contained the definitions of the columns in the system (one row per column), including the columns of SYS.COLUMNS itself. A981. Each such definition included the name of the column's table, the name of the column, and the column's number. A981; A3773. Chang also disclosed that rows in the system each had a record i.d. A3767(7:51-8:3); A225, A232; A357, A361-62. The Board thus correctly concluded that each column and each row of SYS.COLUMNS had an array of bits that identified the row or the column and thus had an OID. A23; A102-03. The Board also correctly found that the rows of SYS.COLUMNS that held the column definitions of that table included the OIDs of the columns in the form of the column names and numbers. A25; A107-08.

Although the Board did not reach SYS.TABLES, the same analysis applied there as well. That table had multiple rows and columns. A992. The columns were named and numbered. A981; A3773. Each row held the name, creator and a packed descriptor of one table in the system, including SYS.TABLES itself. A981; A3774. And the packed descriptor contained the definition of every column

in the referenced table and every index of data in the table. A3774; A992-93. Because the columns were each identified with a name and number and the rows included the table name and had record i.d.'s (OIDs), claim 31's requirements of logical rows and logical columns were satisfied. The self-referentiality requirement of claim 31 also was met because the packed descriptor in the row for SYS.TABLES included the column definition (including column name and number) of all the columns in that table. A220; A230-31; A353, A365-66.

Thus, both SYS.TABLES and SYS.COLUMNS anticipated claim 31 of each patent.

B. The same tables also anticipated dependent claims 32

Claim 32 of each patent adds the requirement that one column be "defin[ed]" "to contain information for enabling determination of OIDs from text entry." Both SYS.TABLES and SYS.COLUMNS satisfied that limitation.

In SYS.TABLES, column 3 of the table was defined to contain packed descriptors for all the tables of the system. A981; A3774. Each row of SYS.TABLES corresponded to a given table and held the packed descriptor for that table. A981; A3774. The row corresponding to SYS.TABLES held the packed descriptor for the SYS.TABLES table. The row corresponding to the EMPLOYEES table held a packed descriptor of the EMPLOYEES table.

As shown in Chang's Figure 6, each packed descriptor included column OIDs for its associated table:

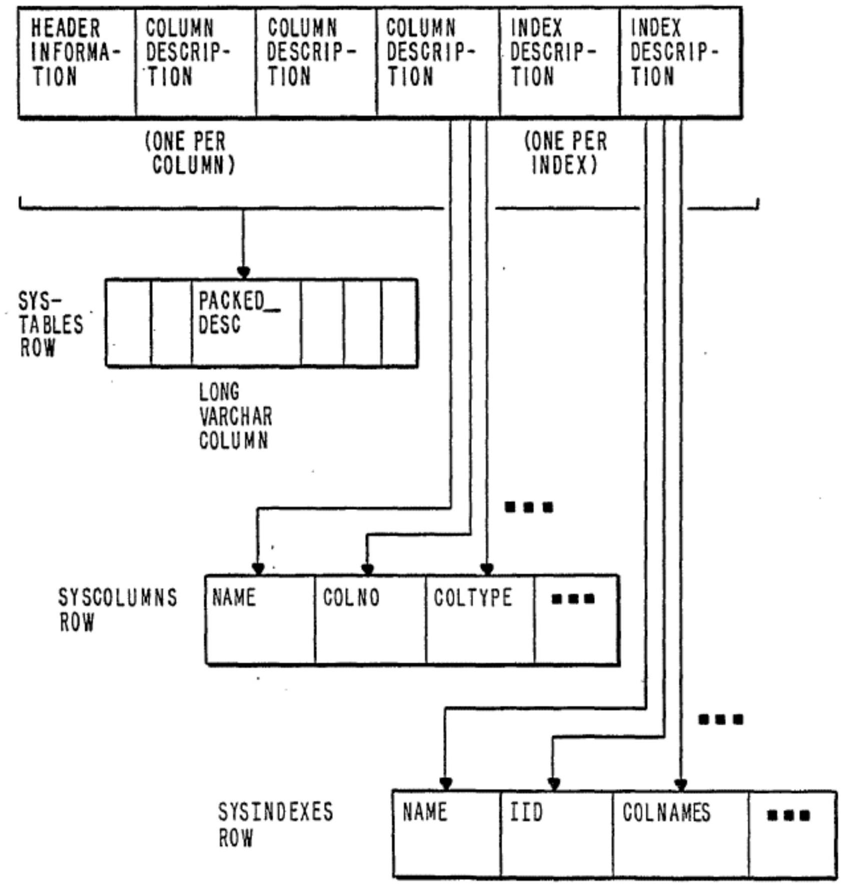


FIG. 6

GENERIC PACKED DESCRIPTION COLUMN ENTRY IN GIVEN SYS. TABLES ROW

A3774. Specifically, each packed descriptor included a definition for each column in the table described by that descriptor, and the definition in turn incorporated the defined column's name and number, among other defining characteristics. As reflected in Figure 6, those column definitions (together with the column names and numbers) were redundantly stored in the rows of the SYS.COLUMNS table. In addition, Figure 6 shows that the packed descriptor of a table in SYS.TABLES

included the definition of all existing indexes of data in the table. An index definition included OIDs for the index in the form of the index name, its index i.d., and the names of all columns covered by the index. A3773-74.

As Chang explained, the purpose of the packed descriptor in SYS.TABLES was to enable obtaining the entire schema/definition for a table and its associated indexes with just one input/output operation. A3763. Whenever a user sought to retrieve a record from the system, the user would type a query on the keyboard and the system would perform a validation check on the query against the definitions for the table where the record resided and any definitions of indexes that might be available to locate it within the table. A3763(Abstract), A3764(1:27-36). Chang described this as part of "compiling" the query command, meaning translating the query from something the user understands, A3764(1:36-39), into a format capable of direct use by the system, A3764(1:40-42). Without the packed descriptor, each column and index definition would have to be obtained sequentially from SYS.COLUMNS, one input/output operation at a time. Chang expressly taught that column definitions could be acquired and determined through either SYS.TABLES or SYS.COLUMNS because the information was redundantly contained in both tables. A3765(4:17-25).

As Microsoft explained to the Board and explains again below, Chang's acquisition and use of column and index definitions when validating queries entered through keyboard text entry satisfied claims 32 in two ways.

1. Column OIDs were determined in response to text entry of query statements by retrieving column definitions in SYS.COLUMNS or SYS.TABLES

Like relational databases generally, Chang relied on conventional retrieval of records in a table through keyboard queries entered in the form of SQL statements. Such statements required entry of text such as the name of the table containing the record of interest and other criteria delimiting the query's purpose. Once entered, the query was compiled and converted from the user's text-based language into an internal digital format used by the system. A3764(1:36-42).

Compilation in Chang included a validation of the query parameters against the schema or column definitions of the table containing the record or records of interest. A3764(1:42-48, 2:33-42), A3771(15:19-35). Those definitions incorporated column OIDs to be checked, such as column name and column number. A3764(1:42-48), A3774(Fig. 6). The column definitions were redundantly stored in SYS.COLUMNS, with one column definition per row, and in SYS.TABLES, with all column definitions for a table in a single packed descriptor. A3765(4:3-9), A3774(Fig. 6). Whichever source was accessed, the column OIDs were neces-

sarily determined and retrieved in response to the text entry of a query statement as part of the validation procedure. A3764(1:42-48, 2:33-42), A3771(15:19-35).

The column OIDs accessed and determined in Chang could be OIDs for columns in tables other than SYS.COLUMNS or SYS.TABLES. That would happen, for example, when a query sought information stored in the EMPLOYEES table. Significantly, however, the column OIDs accessed and determined in Chang also applied to columns in SYS.COLUMNS or SYS.TABLES themselves. Chang specifically taught that the SYS.COLUMNS table and the SYS.TABLES table could each be queried. A3763(Abstract), A3765(4:3-9). Thus, a user querying SYS.TABLES would cause the column OIDs for that table to be determined from the packed descriptor in the row corresponding to that table as part of query validation. Similarly, a user querying SYS.COLUMNS would cause the column OIDs for that table to be determined from the SYS.COLUMNS rows that stored column definitions as part of query validation. Chang made no distinction between the system catalogs (SYS.TABLES and SYS.COLUMNS) and other tables in describing how the validation process proceeded. A3771(15:25-35).

2. Alternatively, row OIDs were determined through columns defined to contain index information that facilitated searching

Chang anticipated claims 32 in a second way as well: through the index descriptions in the packed descriptors of SYS.TABLES. Microsoft showed, with-

out contradiction, that relational databases such as Chang inherently enabled records/rows of a table to be identified by searching on a "primary key column"—i.e., a column (or set of columns taken together) in which the value for each row would be unique. A234; A371; A1579-80 ¶ 183; A2705-06 ¶ 199. Chang demonstrated adherence to that well-known feature by showing a column in the exemplar EMPLOYEES table in which the record row for each employee included a column with a unique employee number:

10

EMPLOYEE TABLE ("EMP")

EMP NO	EMP NAME	SALARY		DEPT NO
⋮	⋮	⋮	— — —	
38362	DOE	$ 75,000		A6A
40000	HOWIE	100,000		525
10000	SMITH	30,000		B25

FIG. 1

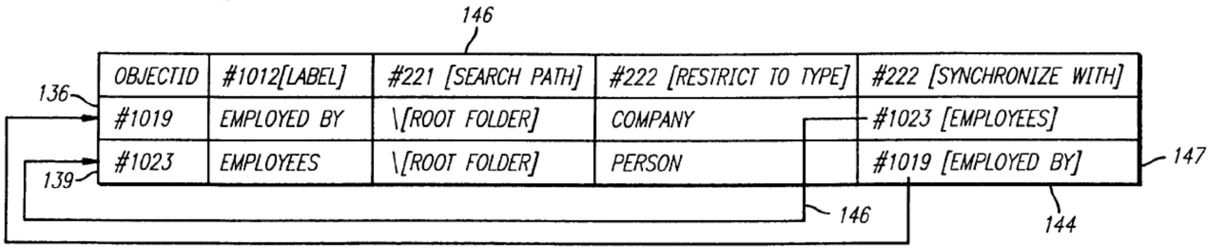
A3773. Conventionally, and as suggested in Chang, such primary key columns were indexed to speed up searching to identify the key and record of interest. A1522-23 ¶ 19; A2642; A8097(165:7-13). Some of the columns identified for searching in the exemplar table were shown in Chang's Figure 4. A3773. As further shown in Chang's Figure 6, the packed descriptor column in SYS.TABLES included identifiers for the columns of a table that were indexed so that they could be used in finding the record matching the user's query. A3774.

C. The Board's decision upholding claim 32 turned on legal errors in determining claim scope

The Board's final written decision criticized Microsoft's position on the packed descriptor in SYS.TABLES and the column definitions in SYS.COLUMNS for not explaining "how OID determination by text *searching* as recited in claim 32 would be conducted on such stored information and definitions in these particular tables." A27 (emphasis added); A112 (requiring a search function). The Board's analysis was mistaken because claim 32 does not require text searching. The claim language in both patents recites "determination of OIDs from text *entry*," not determination of OIDs from text *searching*. Although text entry often precedes text searching, text entry also precedes other database functions that do not involve searching, such as adding new records. A3767(7:39-40) (showing adding a row to a table via a text command); '604(8:59-60). The ordinary meaning of "text entry" is therefore broader than text searching, so the Board's interpretation was too narrow.

The Board may have been misled into restricting claims 32 to text searching because of a preferred embodiment in the patents illustrated in the table of Figure 5 (below) and the corresponding flow chart of Figure 6:

FIG. 5



’604(Fig. 5). Figure 5’s table included a column (146) defined to store a search path consisting of no more than a folder (“\[*ROOT FOLDER]*”). In this embodiment, a user entered a text search on a column. A search within the indicated folder was then conducted to find any rows that match the search criteria. If rows were returned, their OIDs, which were stored in the first column of the table, were also returned.

But the searching aspect of Figure 5 is claimed in dependent claim *33*:

> 33. The method of claim 31 wherein one of said logical columns contains information including a *search path* that references a folder, said folder including a group of logical rows of a similar type.

(emphasis added). Claim 32 of each patent omits any mention of searching, and the file history confirms that the omission of searching from claim 32 in both patents was deliberate. Both patents are continuations from a parent patent that had a dependent claim identical to claim 32 of the ’604 patent, except that it referred to “one of said columns” instead of “one of said logical columns.”

’604(23:32-35). During prosecution of the parent, the dependent claim was changed from requiring

> column information defining said column includes information for searching said column

to

> column information defines one of said columns to contain information for enabling determination of OIDs from text entry.

A4976. A similar change from requiring “searching” to requiring “text entry” happened between the application claims and those that issued with the ’604 and ’775 patents. A4007 (claim 30), A4144, A4153 (claim 88); A4385 (claim 30), A4521, A4530 (claim 88). The omission of “searching” from claim 32 in both patents shows that claim 32, unlike 33, extends to text entry whether or not such entry is coupled with text searching.

Moreover, restricting claims 32 to text searching would improperly exclude the embodiment of Figure 3, in which columns are defined to permit OIDs to be determined from text entry *without* also requiring searching. As its header label confirms, Figure 3’s first column (120) is defined to hold OIDs for all rows of the table. Upon text entry to create a new row, the OID for the row is determined and stored in that column:

FIG. 3

	120	122	124		126		100
108	OBJECT ID	TYPE [# 101]	[#1012] LABEL	ADDRESS [#1013]	EMPLOYED BY [#1019]	TITLE [#1033]	AUTHOR [#1032]
110	#1100	#1020 [COMPANY]	DEXIS	117 EAST COLORADO		N/A	N/A
138	#1101	#1010 [PERSON]	SCOTT WLASCHIN		#1100 [DEXIS]	N/A	N/A
	#1118	#1030 [BOOK]					#1122
	#1122	#1050 [MEMO]					#1122
	#1127	#1060 [DOCUMENT]		C:\WORD\PROJ.DOC		PROJECT PLAN	#1101
136	#1019	# 210 [FIELD]	EMPLOYED BY				
135	# 210	# 111 {TYPE]	COLUMN				
140	# 111	# 111 [TYPE]	TYPE				

130 134 132 133

’604(6:42-43, 8:7-8). Because the OID is not determined based on searching, the Board’s unduly narrow interpretation of claims 32 would exclude Figure 3. In contrast, the broadest reasonable interpretation, in which text entry need not precede text searching, encompasses both Figure 3 and Figure 5. *See Creative Integrated Sys., Inc. v. Nintendo of Am., Inc.*, 526 F. App’x 927, 934 (Fed. Cir. 2013). And under that construction, the Board’s distinction of Chang evaporates.

The Board repeated its “searching” error and made further mistakes of claim scope in denying Microsoft’s petitions for rehearing of claims 32. In those requests, Microsoft argued that even if a searching requirement were imported into claims 32, the Board overlooked that the packed descriptor column in SYS.TABLES was defined to include indexing information used in searching.

A1348; A1367. The Board responded that Microsoft had not shown that Chang determined OIDs by searching using the index information in the packed descriptor "alone." A1359, A1361, A1363-64; A1378, A1380, A1382-83. But the Board went too far in requiring the recited column information to be sufficient "alone" to enable determination of OIDs from text entry. Even Figure 5's searching embodiment does not show column information of that caliber. The relevant column information for searching in Figure 5 consists solely of the name of a file folder "\[ROOT FOLDER]." Although having the folder name would help shorten the searching process, it would not be enough to complete it. Chang similarly taught that the packed descriptor of the SYS.TABLES table included index descriptions that identified which columns had been indexed in order to facilitate the searching process. A234; A370-71; A1580; A2706.

Finally, the Board's denial of rehearing relied on a deposition question and answer from Microsoft's expert, Dr. Hosking, which the Board interpreted as suggesting that Dr. Hosking could not "confirm that any single table in Chang enabled determination of OID's from text entry." A1361; A1380. There are multiple problems with the Board's reliance on that testimony excerpt.

First, the question was restricted to "the eight paragraphs in the section for claim two" in Dr. Hosking's declaration and asked whether those eight paragraphs

showed a single table with a single column with information for enabling "determin[ing] OIDs in that table" from text entry:

> Q: My question in the eight paragraphs in the section for claim two, can you point to a single table – a single logical table in which logical column information exists that defines one of the logical columns in that table to contain information for enabling determination of OIDs in that table from text entry?
>
> …
>
> A: Yeah. I think my answer was no.

A1361; A8129-30(197:22-198:7).

The question by its terms excluded much of what Dr. Hosking wrote about Chang in view of claims 31 and 32, including paragraphs earlier in the declaration that detailed how each of SYS.TABLES and SYS.COLUMNS included information that defined the columns of that table and why such column definitions incorporated column OIDs for the same table that were accessed and determined in the validation of queries entered by the user on a keyboard. A1574-77; A2698-2703.

Furthermore, as Dr. Hosking's answer indicated, the subject matter of the question had been discussed earlier in deposition. In fact, in response to the immediately preceding questions, Dr. Hosking had explained that OIDs of SYS.TABLES and SYS.COLUMNS may be comprised of the table name and either column name or number taken together, rather than taken alone. A8030-31(98:13-99:24); 8128(196:2-19). Thus, the information for determining OIDs in

SYS.TABLES or SYS.COLUMNS could be included in more than one column rather than in just one of the columns as the questioning attorney insisted when refocusing the question. The questioner artificially restricted the scope of the question to focus on a single column rather than the combinations of columns presented by Dr. Hosking and taught in Chang. In context and under a proper claim construction, Dr. Hosking's testimony confirmed anticipation of claims 32.

Because claims 32 require text entry, not text entry followed by searching, and because the SYS.TABLES and SYS.COLUMNS tables of Chang each defined columns for the purpose of determining column OIDs from text entry, the Board's determination of no anticipation should be reversed. Alternatively, that ruling should be reversed because the packed descriptor column of SYS.TABLES was defined to include index information used to search for row OIDs, which sufficed even if additional information would be needed to complete the search.

II. The Board's non-invalidity determinations regarding claims 36 and 37 of each patent were based on erroneous claim constructions

Although the Board found claim 31 of each patent anticipated by Chang, it ruled that Microsoft had not proven that dependent claims 36 and 37 were obvious over combinations of Chang with other references. But those determinations must be vacated because they turned on erroneous claim constructions.

A. The Board mistakenly required the "different logical row/record" of claims 36 to be in the same table

Claim 36 of the '604 patent recites:

> The method of claim 31 wherein:
>
> at least one of said plurality of logical rows includes information defining the type of *a different logical row*; and
>
> at least one of said plurality of logical rows includes a logical cell that contains a pointer to said logical row including logical row type information.

(emphasis added). Claim 36 of the '775 patent is essentially similar.

Microsoft contended that both claims 36 were obvious in light of Chang and Goldberg. In particular, Microsoft asserted that Chang satisfied the first clause because the packed description in SYS.TABLES included a logical row defining the type of another logical row: a row in the related SYS.COLUMNS table. A1619 ¶ 310; A2748-49 ¶ 336. The Board agreed with Enfish, however, that Chang did not satisfy the "different logical row" limitation because the defined row in SYS.COLUMNS was not in the same table as the defining row in SYS.TABLES. This was the Board's sole ground for distinguishing Chang, and the ruling turned on a matter of claim construction: whether the defining row and the defined rows must appear in the same table. A43-44; A123-24.

The Board's construction was wrong because it did not reflect the broadest reasonable interpretation of the claim language. In effect, the Board rewrote

> at least one of said plurality of logical rows includes information defining the type of a different logical row

as

> at least one of said plurality of logical rows includes information defining the type of a different ~~logical row~~ *one of said plurality of logical rows*.

(And likewise for ’775 claim 36, with “record” instead of “logical row.”) But Enfish did not write the claims that way. When Enfish wanted to refer to “one of said plurality of logical rows/records,” it said so. Here, it merely required “at least one of said plurality of logical rows/records” to define the type of “a different logical row/record” without specifying that the “different” row/record had to be in the same table.

Nothing in the specification dictates that the “different logical row/record” must be in the same table. The term appears only in the claims. Figure 3 does show a row defining the type of another row in the same table (*see* rows 135 and 140), but it is just a preferred embodiment and not coextensive with the claims. *See also* ’604(12:16-20) (describing Figure 11 in which “list 250 is shown separately from the table 100” but noting that in the preferred embodiment the list is part of that table).

Moreover, during prosecution of the parent patent with the identical written description, the examiner rejected claims with nearly identical wording (“at least one of said plurality of rows includes information defining the type of a different

row") over a combination including U.S. Patent No. 5,421,012 to Khoyi in which a row defined the type of another row in a *different* table. *See* A4957-58 (proposed claims 6, 34); A4960 (rejecting claim 6 in view of Khoyi at 10:41-11:2, which in turn referred to an "object catalog" in which row entries of the Link Table defined objects in the separate Object Table shown in Khoyi's Figure 5).[2] Enfish acquiesced in that rejection, dropping claims 6 and 34. *See* A4984.

The broadest reasonable interpretation of "a different logical row/record" is thus "a different logical row/record," not "a different logical row/record *within the same table*." Chang plainly disclosed a logical row/record defining a different logical row/record. Because the sole basis for the Board's finding that claims 36 were non-obvious over Chang + Goldberg was its mistaken view that the defining and defined rows had to be in the same table, the Board's ruling should be vacated and the case should be remanded for further consideration of those claims.[3]

B. The Board mistakenly required the "pointers" of claims 37 to refer to addresses rather than locations

Claim 37 of the '604 patent recites:

---

[2] Although the examiner's rejection based on Khoyi is in the record, Khoyi itself is not. But this Court can and should take judicial notice of the contents of Khoyi, which is a U.S. patent and thus a public record.

[3] Enfish argued that claims 36 were not obvious for other reasons, but the Board did not reach those arguments in its final written decision. Microsoft recognizes that the Board must address those issues in the first instance.

> The method of claim 31 wherein at least one of said logical columns defines logical cells that include a plurality of *pointers* to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

(emphasis added). Claim 37 of the '775 patent is essentially similar.

Microsoft contended that claims 37 were obvious over the combination of Chang and Anderson, which described Quattro Pro spreadsheet tables in which cells could contain formulas pointing to values in other cells. In Anderson's Figure 4J, for example, cell B7 contained a formula pointing to the values of cells B6 and A7. *See* A251; A385; A1622-23 ¶¶ 322-324; A2751-53 ¶¶ 347-350. In instituting review, the Board rejected Enfish's argument that such references to other cells could not qualify as "pointers." A831; A893-94; *see also* A823-24; A886 (finding no need to define "pointer"). But in its final written decisions, the Board reversed course, held that a "pointer" had to refer to an "address where another object resides," and ruled that Microsoft had not shown that a reference by the formula in one cell to a variable value of another cell constituted "stor[ing] an address to a location where an object resides in the database." A45-46; A125-26. This determination depended on the Board's claim construction requiring a "pointer" to refer to the *address* of an object in the database (memory), rather than simply the location of another cell in the table.

That construction was wrong because it did not reflect the broadest reasonable interpretation of the term "pointer." To begin with, claims 37 nowhere refer to an "address." Nor does the rest of specification: its only mentions of "addresses" describe *postal* addresses of employees in an exemplar personnel database. *See* '604(Figs. 3, 9, 11). Furthermore, the specification makes clear that a "pointer" need *not* refer to an address in the database. Although the specification does not define "pointer," it does explain that the OIDs described in the patent "are pointers to other records." '604(8:51-52); *see also* '604(6:47-49) ("the OID's for both rows and columns may be used as pointers"). Similarly, the patent identifies a cell identification number comprised of two OIDs as another type of "pointer." '604(12:22-26). But the OIDs described in the patent are not addresses of locations in memory; they are simply numbers consisting of a timestamp, a session ID, and a tiebreaker. '604(8:34-37). Moreover, the patent explains that OIDs are useful as pointers because "[a]n efficient query can use these OID's to go directly to another record, rather than searching through columns." '604(8:51-54). Accordingly, a "pointer" need not include a specific memory address; instead, it just needs to provide a shortcut to another record or cell. Indeed, the Board recognized this in its decisions to institute, which explained that OIDs "act as direct pointers to other records, which eliminates the necessity for a query to

search through unrelated columns, rows, and cells for additional related data." A808 (citing '604(8:51-54)); A869 (citing '775(8:59-63)).

In its final written decisions, the Board recognized that the patents describe using OIDs as pointers to objects, A45-46; A126, but it ignored that the OIDs do not provide addresses, just shortcuts to other records/cells. The Board also relied on extrinsic evidence: textbooks on the C++ and Java programming languages cited by Enfish's expert. A45; A125-26. But those textbooks proved nothing. Two suggested that a pointer in C++ stores the address where another object resides, A7882-83 ¶¶ 208-209, but this invention is not limited to any particular programming language. The third simply said that in Java, a "reference type" "can be thought of as a *reference to* (or a *pointer to*) an object of the appropriate type." A7883 ¶ 210. This sentence merely equated a "pointer" with a "reference"; it did not require pointing to a specific *address* in a database.

Under the broadest reasonable interpretation, the "pointers" of claims 37 need only point (refer to) another record/cell. And under that definition, the spreadsheet formulas in Anderson plainly included pointers to other cells. The

Board's ruling should therefore be vacated and the case remanded for further consideration of these claims.[4]

III. Claim 43 of the '604 patent and claim 42 of the '775 patent were obvious over Chang + Horn

Claim 42 of the '604 patent is similar to claim 31 except that (a) it omits claim 31's requirement of equal OIDs; (b) it requires a logical row to "contain[ ] a logical cell that contains a pointer to a different logical row"; and (c) it further requires "searching said table for said pointer." The Board correctly found that Chang combined with Horn rendered claim 42 obvious. As discussed below, however, the Board erred in finding that the same combination of references did not invalidate dependent claim 43, which adds a requirement of column defining cells that include "a plurality of pointers to other logical columns within the same record that contain defined values." The Board reasoned that Microsoft was relying on two tables from Chang rather than one, but that was not so: Microsoft relied on SYS.TABLES alone.

For its part, claim 41 of the '775 patent is similar to claim 31 of that patent. Again, the Board correctly ruled that claim 41 was invalid (this time over Chang alone), but it erred in refusing to invalidate dependent claim 42, which adds a

---

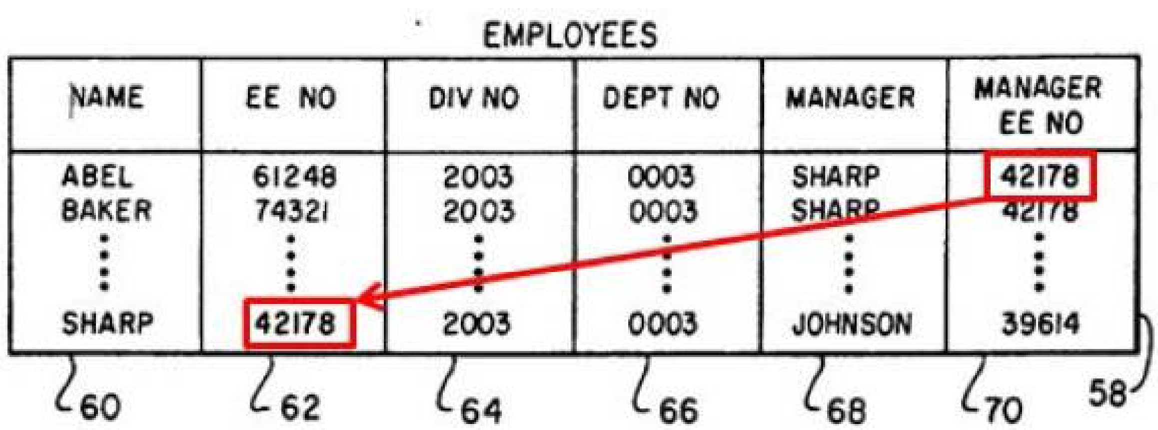
[4] Enfish argued that claims 37 were non-obvious for other reasons not reached in the Board's final written decision. Microsoft recognizes that the Board must address those arguments in the first instance.

limitation paralleling ’604 claim 43. This time, however, the Board’s error was one of claim construction: it relied on the overly narrow construction of “pointer” that it adopted in connection with claims 37.

The Board’s rulings on these claims should therefore be vacated and the cases remanded for further consideration of Microsoft’s actual arguments under the correct claim construction.

A. The Board correctly found that claim 42 of the ’604 patent was obvious over Chang + the row-to-row pointer of Horn

Apart from incorporating arguments regarding claims 31, Enfish’s defense of ’604 claim 42 rested on its assertion that Chang and Horn did not disclose row-to-row pointers. A1006-08. Microsoft showed, however, that Horn disclosed that one row within a table could have a pointer to another row within that table. The annotated figure below illustrates exactly that:



A1610 (annotating a figure from Horn at A3782). Furthermore, Microsoft's expert, Dr. Hosking, testified that row-to-row pointers were a common tool for database designers and that Horn itself did not treat its row-to-row pointers as an innovation or a tool limited to employee tables. A1610 ¶¶ 270-271. In response, Enfish urged that the row-to-row teaching in Horn should be confined to employee tables and that a skilled artisan would not have used that technique with Chang's system catalog tables. A1006-07. Enfish also argued, as it had for claims 37, that "pointers" must point to an address where an object resides.

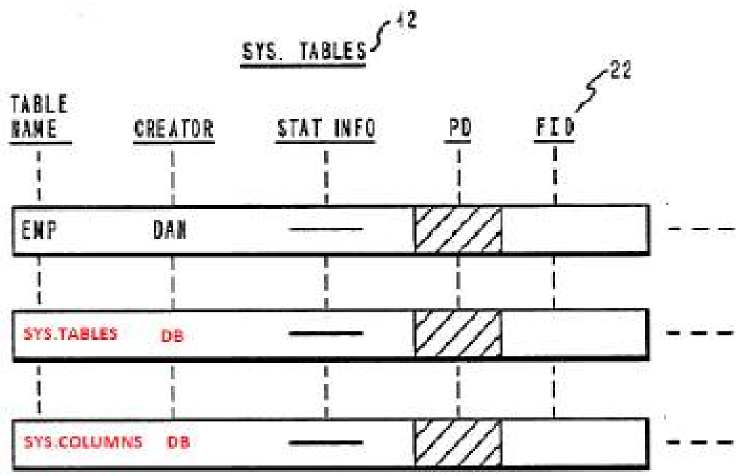
The Board rejected both of Enfish's distinctions. The Board credited Dr. Hosking's testimony and found that the use of row-to-row pointers, as depicted in Horn, was an ordinary design choice that would have been obvious to incorporate in the system tables of Chang. A29. The Board also found that Horn's pointer satisfied even Enfish's narrow construction of "pointer." A30.

B. The Board mistakenly assumed that Microsoft's arguments regarding '604 claim 43 relied on two tables

Although '604 claim 42 focuses on row-to-*row* pointers, the additional limitation of '604 claim 43 requires pointers to other logical *columns* within the same record that contain defined values. Microsoft contended that claim 43 was obvious in light of the SYS.TABLES table in Chang in combination with the row-to-row pointer feature of Horn. A236-41. In rejecting that argument, the Board

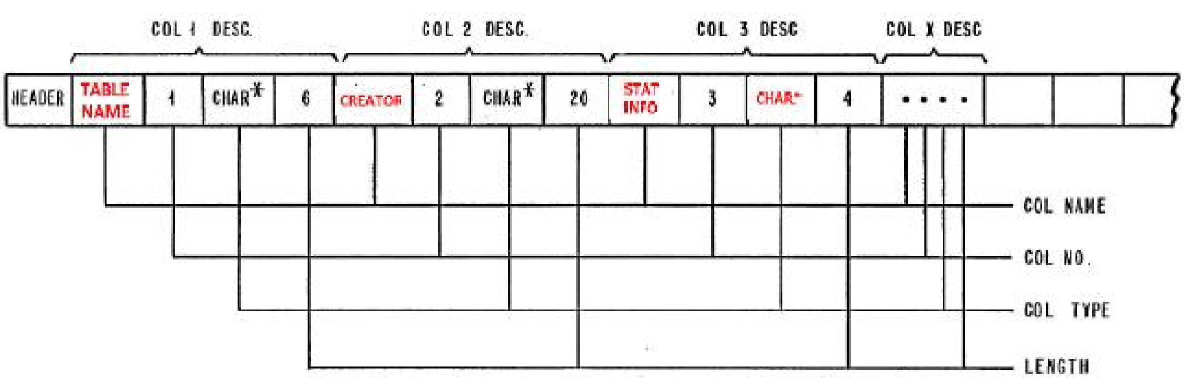
mistakenly assumed that Microsoft was relying on a combination of SYS.TABLES and SYS.COLUMNS when Microsoft actually relied on SYS.TABLES alone.

SYS.TABLES undisputedly included a row for the SYS.TABLES table itself. Indeed, Enfish's own annotated figure showed this:



Annotated FIG. 2

A992. There was also no dispute that this row of SYS.TABLES contained a packed descriptor column with the names and numbers of every column within the table. Enfish's own annotated table showed this too:



Annotated FIG. 7

A993-94 (with COL. NO. line pointing to the numbers 1, 2, and 3). Microsoft established that the SYS.TABLES column numbers stored in the packed descriptor of that table satisfied claim 43's limitation requiring pointers to columns with defined values in that table.

The Board rejected Microsoft's argument on the premise that the column numbers within the SYS.TABLES packed descriptor identified *rows* within the SYS.COLUMNS table. A31. But whether they did so is irrelevant because the additional limitation of claim 43 does not require pointers to *rows*. It requires pointers to "other logical *columns* within the same record," and Microsoft showed that the packed descriptor contained pointers to the *columns* of the SYS.TABLES table. *See* A242 ("Chang further discloses logical pointers (e.g., Col. No. 25) that point to the column's logical placement in a table."). Microsoft further showed that the column numbers stored in the SYS.TABLES table identified the locations of the columns within that table. *See* A7855-56 ¶ 136 (indicating that column number 1 was the first column).

Because the Board addressed a false issue (pointers to rows) rather than addressing Microsoft's actual argument regarding pointers to columns, the Board's determination regarding '604 claim 43 should be vacated with instructions to reconsider Microsoft's actual argument on remand.

C. The Board's ruling on '775 claim 42 should be vacated because the Board again relied on an overly narrow construction of "pointer"

The Board agreed with Microsoft that Chang anticipated independent claim 41 of the '775 patent. A107-11. The Board held, however, that Microsoft failed to show that dependent claim 42 was obvious over the combination of Chang + Horn and Anderson. A126. The Board's sole reason was that Chang and Anderson did not disclose "pointers" under the Board's construction requiring "pointers" to store an "address" to a location where an object resides. *Id.*

Microsoft explained above, in connection with claim 37, why the Board's construction of "pointer" was overly narrow and why "pointers" include variables identifying locations elsewhere in a table. Chang and Anderson undeniably disclosed "pointers" when the term is construed according to its broadest reasonable interpretation. Anderson included pointers from one cell to another in the same table. The pointers consisted of a target cell's row and column number. A3830-31. The Court should therefore order reconsideration of '775 claim 42 under the correct construction of "pointer."

IV. The indexing claims dependent from claim 54 of both patents were obvious over the Visual Basic manual + Salton

Although the Board correctly found independent claim 54 of each patent anticipated by the Visual Basic manual, the Board refused to invalidate claims 55, 56, and 60 of each patent on grounds that Microsoft failed to prove it would have

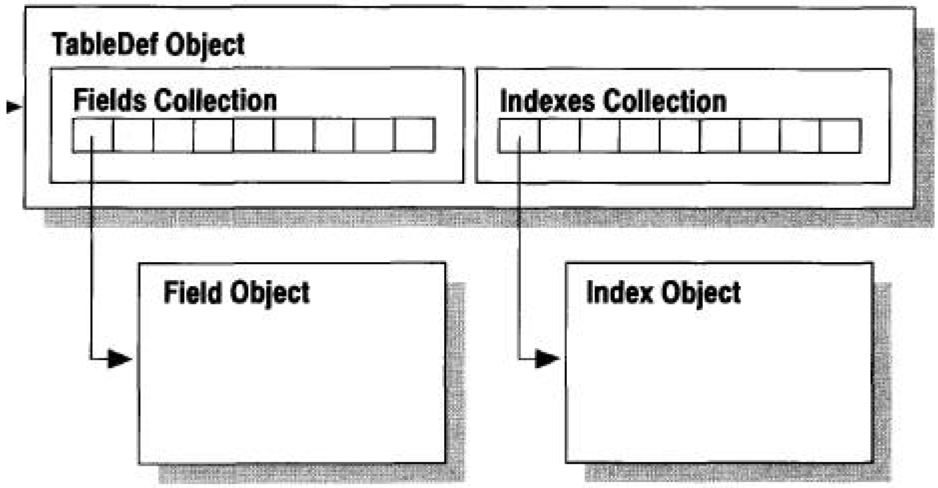
been obvious to combine the Visual Basic manual with discussions of indexing in Salton's textbook *Introduction to Modern Information and Retrieval*. The Board's determination cannot stand because the Board misapplied the law on motivation to combine and Enfish's only counterarguments were based on an incorrect claim construction.

A. The Board correctly concluded that the Visual Basic manual anticipated independent claim 54

Claim 54 of each patent recites a method of storing and retrieving data comprising two steps: (1) configuring memory according to a logical table containing rows/records and columns/attribute sets with OIDs and at least one cell with a pointer to an index record; and (2) indexing the data in the table. The Board correctly found that the Visual Basic manual satisfied these requirements.

The Visual Basic manual described a Table Object that was a table including rows/records with primary keys acting as row OIDs and columns/fields whose ordinal values served as column OIDs. A7408; A7566; A7275; A7035; A6614; A1548 ¶ 87; A2666 ¶ 80. The manual also described indexing the data within the Table Object by creating an Index object. A7564 ("Adding an index to your database can increase the speed with which you get access to information …. Each index in the table is defined by an Index object; the details of each Index object are kept in the Indexes collection."). Finally, the Table Object, as defined by the

TableDef object, included a cell with a pointer to the index record, as shown in Figure 20.3:



A6614; A7130.

B. The Board's conclusion that dependent claims 55 were not obvious relied on a strict motivation-to-combine standard inconsistent with *KSR*'s flexible approach

Claim 55 of each patent adds two further requirements to the indexing step of claim 54: (1) "searching said table for a key word"; and (2) "creating an index record for said key word, said index record having one or more pointers to a logical cell in said table that contains said key word." Enfish did not dispute that the Visual Basic manual disclosed searching the Table Object for a keyword using Seek and Find methods. *See* A7353; A7049. Enfish did dispute whether the cited art disclosed the second limitation.

The second limitation recites a conventional index-creation technique. The idea of creating an index associating a keyword with locations where that keyword

is found is a ubiquitous concept that existed well before the computer and database era. For example, books have long included indexes in the form of an alphabetized list of keywords along with numbers of pages where those words appear in the text. The indexing technique described and claimed in Enfish's patent is essentially the same as this centuries-old indexing approach. Indeed, rather than describing its indexing approach as new, Enfish's patents recognized that creating indexes, even elaborate full-text indexes, was already standard practice. '604(13:45-47) ("In full text extraction, every word is indexed, which is typical for standard text retrieval systems.").

Because the Visual Basic manual was a product manual for advanced programmers, it described using indexes but did not provide low-level implementation details about how to create them in the first place. To provide that supplemental information, Microsoft presented Salton, a well-known textbook describing basic principles and techniques for indexing. Salton plainly disclosed creating an index record for a keyword with one or more pointers to a table cell containing the keyword. A7654-55. For example, Salton referred to files where a single word is mapped to multiple data fields as "inverted files" and noted that they were "used in almost every commercially available information retrieval system." A7655. The portions of Salton on which Microsoft relied appeared in the first chapter of the

book, entitled "Information Retrieval: An Introduction," confirming that Salton was describing basic, well-known techniques.

The Board refused to find claims 55 obvious not because Salton's disclosure of the claimed index record was insufficient, but because the Board thought Microsoft had not provided a sufficient motivation for a person of ordinary skill in the art to combine the teachings of Salton and the Visual Basic manual. A76-78; A154-55. But Microsoft presented ample evidence of a motivation to combine, and in finding the evidence insufficient the Board relied on an overly strict standard reminiscent of the one rejected in *KSR International Co. v. Teleflex Inc*., 550 U.S. 398 (2007).

Under *KSR*, obviousness is resolved by an "expansive and flexible approach," animated by the principle that a "patent for a combination which only unites old elements with no change in their respective functions … obviously withdraws what is already known into the field of its monopoly and diminishes the resources available to skillful men." 550 U.S. at 415-16. The inquiry is broad, and includes (1) determining the scope and content of the prior art, (2) ascertaining the differences between the prior art and the claims at issue, (3) resolving the level of ordinary skill in the art, and (4) accounting for any secondary considerations. *Id*. at 406. Although it is important to identify a reason one of ordinary skill would have combined prior-art elements as recited in the claim to guard against hindsight bias,

the analysis should not be formulaic. The obviousness inquiry under *KSR* requires considering not only the prior art itself, but also common knowledge and common sense. *Randall Mfg. v. Rea*, 733 F.3d 1355, 1362 (Fed. Cir. 2013); *see also In re Taylor Made Golf Co*., 589 F. App'x 967, 971 (Fed. Cir. 2014) ("[A]s part of the obviousness analysis, the prior art must be viewed in the context of what was generally known in the art at the time of the invention.")

Here, the evidence presented by Microsoft presented a compelling motivation to combine the Visual Basic manual with Salton. As noted above, the Visual Basic manual was a product manual for advanced users that discussed indexing at a higher level but did not delve into all possible implementation details. As a matter of common sense, a person of ordinary skill in the art who wanted more detail about indexes would have turned to a textbook or similar reference work devoted to describing them. *See*, *e.g.*, *DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F.3d 1356, 1360 (Fed. Cir. 2006) ("textbooks or treatises may include basic principles unlikely to be restated in cited references"); *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1262 (Fed. Cir. 2007) (finding it obvious to substitute conventional multiplexer circuit described in textbook for multiplexer described in different prior-art reference); *cf. Wyers v. Master Lock Co.*, 616 F.3d 1231, 1241 (Fed. Cir. 2010) (common sense to combine barbell-shaped hitch pin lock with similarly designed hitch from another reference).

As Enfish's expert admitted, Salton was a well-known textbook on indexing, A7897-98 ¶ 242, and thus a logical resource for supplemental information. And as Dr. Hosking testified, "VB3 and Salton address[ed] the same technical issues and disclose[d] closely related subject matters. A person of ordinary skill in the art would [have] be[en] motivated to combine Salton's disclosure of manual indexing with the indexing system disclosed in VB3." A1651-52; A2782; *see also* A1657 ¶ 442; A2787 ¶ 468 ("One having ordinary skill in the art would know to combine what's described in Salton with what's described in VB3 because they both deal with solving the problem of locating key words in a database.").

Referring to the inverted index described in Salton to supplement the tables described in the Visual Basic manual was common sense and required no great technological leap. Salton introduced inverted indexes as one of three "simple file structures" in the introductory chapter of the book, explaining that inverted indexes were a common, beneficial way to increase the efficiency of data retrieval that was "used in almost every commercially available information retrieval system." A7655. Salton thus taught expressly that inverted indexes could be and were used with a wide variety of databases. The structure of such indexes is simple, and Enfish has never suggested that it would have been difficult for a skilled artisan to apply them to the teachings of an ordinary information storage and retrieval system like the one described in the Visual Basic manual.

In finding insufficient evidence of a motivation to combine, the Board deemed Microsoft's contentions insufficient because they were supposedly "based on solely the references teaching the same techniques." A77. The Board demanded "expert testimony that the combinations of prior art references are of familiar elements according to known methods that yield no more than predictable results," A79. But expert testimony need not parrot *KSR*, and the Board disregarded *KSR's* admonition that obviousness inquiries must avoid "rigid and mandatory formulas." 550 U.S. at 419; *see also Belden Inc. v. Berk-Tek LLC*, Nos. 2014-1575 & -1576, 2015 WL 6756451, at *11 (Fed. Cir. Nov. 5, 2015) ("No rule requires a Petition to be accompanied by any declaration, let alone one from an expert guiding the Board as to how it should read prior art."). As this Court has recognized, the motivation to combine need not be explicit, but instead may be found in the nature of the subject matter or derived from logic, judgment, or common sense. *See*, *e.g.*, *Perfect Web Techs., Inc. v. InfoUSA, Inc.*, 587 F.3d 1324, 1329 (Fed. Cir. 2009) (Obviousness analysis "may include recourse to logic, judgment, and common sense available to the person of ordinary skill that do not necessarily require explication in any reference or expert opinion."); *Leapfrog Enters., Inc. v. Fisher-Price, Inc.*, 485 F.3d 1157, 1161 (Fed. Cir. 2007) ("the common sense of those skilled in the art" can be sufficient to "demonstrate[ ] … why some combinations would have been obvious where others would not"); *W. Union Co. v.*

MoneyGram Payment Sys., Inc., 626 F.3d 1361, 1372 (Fed. Cir. 2010) (dependent claims "add[ing] only trivial improvements that would have been a matter of common sense to one of ordinary skill in the art" were obvious).

Following the framework laid out in *Graham v. John Deere Co*., 383 U.S. 1 (1966), Microsoft identified the element of claims 55 that the Visual Basic manual lacked (creating index records for keywords with pointers to where those keyword are located) and demonstrated that it was simply an old element, well-understood by those of ordinary skill in the art, that could be applied with no change in its function. It was obvious for a skilled artisan to turn to a standard text on indexing for more information about how to implement the indexing function discussed in the Visual Basic manual. The final written decisions erred by requiring more. *Cf*. A858; A946-47 (decision to institute recognizing that "Microsoft has set forth a sufficient articulated reason with a rational underpinning to support obviousness" and that "Enfish has not explained sufficiently why Salton teaches away from the combination set forth by Microsoft").

In a final paragraph, the Board observed that Enfish's expert, Dr. Jagadish, disagreed with Dr. Hosking's assessment of the motivation to combine. A78; A155. To the extent the Board credited Dr. Jagadish, it erred. Dr. Jagadish contended that Salton taught away from full-text indexing, defeating the motivation to combine. A7895-98 ¶¶ 239, 242. But nothing in claims 55 requires that all

words in the table be indexed, and the specification described column indexing, automatic analysis indexing, and manual selection indexing in addition to full-text indexing. ’604(13:40-54). It is improper to import limitations from one preferred embodiment into claims that have no language limiting them to particular embodiments. *See*, *e.g.*, *Falana v. Kent State Univ.*, 669 F.3d 1349, 1354-55 (Fed. Cir. 2012) (reversing construction limiting claim to one embodiment: “[i]t is the *claims*, not the written description, which define the scope of the patent right”). Salton disclosed numerous kinds of indexing that fell within claims 55 and aligned with the examples in Enfish’s specification. A7654 (inverted indexes generally), A7665 (adjacency indexes), A7682-83 (full-text), A7692-93 (manual indexing), A7693 (indexing with synonyms), A7708-12 (automatic indexing).

In any event, even if claims 55 required full-text indexing, Salton did not teach away from doing so. Indeed, a passage on which Dr. Jagadish relied observed that “[t]here are many so-called full-text retrieval systems where the full text of the documents is used for indexing purposes.” A7708; A7897. Dr. Jagadish also cited Salton’s statement that “the computer storage of the full text of documents is expensive and is rarely possible except as a by-product of automatic typesetting operations,” A7897, but the fact Salton warned in 1983 that the storage of the full-text of large libraries of medical or legal documents was expensive did not teach away from using an inverted index as described in earlier chapters of the

book and as claimed in claims 55. *See In re Gurley*, 27 F.3d 551, 553 (Fed. Cir. 1994) (claimed epoxy was obvious in view of reference that disparaged its use because reference nonetheless taught that epoxy was usable for cited purpose).

C. The Board's decisions to uphold claims 56 and 60 repeated the same error

Claim 56 of each patent depends from claim 55 and simply requires using the claimed index record for its intended purpose—to retrieve table cells corresponding to a user's query:

> 56. The method of claim 55 further comprising the steps of:
>
> locating said index record according to the query of a user;
>
> retrieving at least one logical cell in said table pointed to by said located index record.

'604(26:23-28). Once again, the Visual Basic manual provided a general disclosure of the concept but not the low-level details required by the claim. In particular, it described using an index in connection with a user's query and retrieving logical cells according to the query, A7353, but it did not specifically disclose "locating said index record" or retrieving results "pointed to by said located index record." As with claims 55, the manual's high-level disclosure provided a plain motivation to combine it with a reference that taught further details about the techniques generally described in the manual.

Again, Salton was such a reference. Salton described an "inverted index" that contained index records, each with a key term and pointers to the objects that corresponded to that key term. Salton also explained that a user search "requires a search of the index to find the desired term, and hence the associated document reference numbers" and that "the identified documents are selected from the document reference file." A7661-62. There is no substantial difference between storing references to a document and storing references to a logical cell, or between retrieving a document and retrieving a logical cell pointed to by such a reference. The combination of the Visual Basic manual and Salton thus rendered obvious the limitations of claims 56.

Claims 60, which depends directly from claims 54, likewise claims a commonplace and common-sense technique, in this case "indexing external documents." Although the Visual Basic manual disclosed a logical table with indexing, it was a manual meant to instruct programmers how to use the Visual Basic programming framework in general and, as such, provided only a handful of examples. Nevertheless, indexing external documents was an obvious use, and Salton once again demonstrated that this practice was commonplace, within the capabilities of one of ordinary skill in the art, and would have been obvious to try. As noted above, Salton specifically explained that "[t]here are many so-called full-text retrieval systems where the full text of the documents is used for indexing

purposes." A7708. Salton also described indexing external documents as one of the key functions of information retrieval systems:

> Information retrieval is best understood if one remembers that the information being processed consists of documents. In that context, information retrieval deals with the representation, storage, and access to documents or representatives of documents (document surrogates). … The output of an information retrieval system in response to a search request consists of sets of references.

A7644. Salton further explained that indexing external documents by creating an inverted file structure with references to documents corresponding to the index entries was commonplace:

> It was indicated in Chapter 1 that virtually all the commercially available systems are based on inverted file designs. That is, each system logically consists of a document file and one or more auxiliary directories known as the "inverted index." The inverted index contains the allowable indexing terms. For each term an associated list of document reference numbers is included in the index. Each document reference number uniquely specifies a document to which a given term has been assigned. Thus, the retrieval of the documents identified by an arbitrary term requires a search of the index to find the desired term, and hence the associated document reference numbers. Finally the identified documents are selected from the document reference file.

A7661-62. Because Salton confirmed that indexing external documents was a common practice well-understood in the art, it was obvious to combine the Visual Basic manual with Salton to practice every limitation of claims 60.

The Board's final written decision did not provide any analysis specific to claims 56 or 60. Instead, the Board found claims 56 and 60 non-obvious for the same reason as for claims 55: that there was insufficient motivation to combine Salton with the Visual Basic manual. That reasoning was wrong for claim 55, and it was equally wrong for claims 56 and 60. It would have been obvious to combine the logical table and index of the manual with Salton's disclosure of indexing external documents because the manual and Salton both described information retrieval systems; the manual's high-level disclosure of indexing naturally suggested combining it with detailed indexing techniques like those described in Salton; and in light of the prevalence of systems which indexed external documents, it would have been common sense to combine such a system with the Visual Basic information and retrieval system that included logical tables.

CONCLUSION

The Board's non-invalidity rulings regarding dependent claims 32, 36, 37, 55, 56, and 60 of both patents, 43 of the '604 patent, and '42 of the '775 patent should be reversed or vacated.

Respectfully submitted,

PERKINS COIE LLP

by /s/Chad S. Campbell
Chad S. Campbell

Counsel for Microsoft Corporation

Addenda

PTAB Decision in IPR2013-00562

Trials@uspto.gov
571.272.7822
Paper 64
Entered: March 3, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

ENFISH, LLC,
Patent Owner.

Case IPR2013-00562
Patent 6,151,604

Before THOMAS L. GIANNETTI, BRYAN F. MOORE, and SCOTT A. DANIELS, *Administrative Patent Judges.*

DANIELS, *Administrative Patent Judge.*

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.73

I. INTRODUCTION

A. Background

Petitioner Microsoft Corporation ("Microsoft") filed a petition to institute an *inter partes* review of claims 1–15 and 31–45 of U.S. Patent No. 6,151,604 ("the '604 patent"). Paper 1 ("Pet."). We instituted trial for claims 1–15 and 31–45 of the '604 patent on certain grounds of unpatentability alleged in the Petition. Paper 15 ("Decision to Institute" or "Inst. Dec.").

After institution of trial, Patent Owner, Enfish, LLC ("Enfish"), filed a Patent Owner Response ("PO Resp., Paper 32), along with a Declaration by Enfish's Declarant, Dr. H.V. Jagadish ("Jagadish Declaration"). On September 16, 2014, Enfish filed an unopposed motion to correct typographical errors in the Jagadish Declaration (Ex. 2007). We grant Enfish's motion to correct both papers.[1] Microsoft filed a Reply ("Pet. Reply"). Paper 37.

A consolidated hearing for IPR2013-00559, IPR2013-00560, IPR2013-00561, IPR2013-00562, and IPR2013-00563 was held on December 3, 2013. The transcript of the consolidated hearing has been entered into the record. Paper 63. ("Tr.").

We have jurisdiction under 35 U.S.C. § 6(c). This final written decision is issued pursuant to 35 U.S.C. § 318(a).

We conclude that Microsoft has shown by a preponderance of the evidence that claims 31, 41, and 42 of the '604 patent are unpatentable. Microsoft has not shown by a preponderance of the evidence that claims 32–

[1] For ease of reference, throughout we refer to Enfish's corrected Jagadish Declaration (Exhibit 2007), filed on September 16, 2014.

40 and 43–45 are unpatentable. For the reasons discussed below, for claims 1–15 we are unable to reach a determination on the alleged grounds of unpatentability over prior art. Accordingly, we terminate this proceeding with respect to claims 1–15 under 37 C.F.R. § 42.72.

B. Additional Proceedings

In addition to this petition, Microsoft has filed a petition challenging the patentability of claims 16–30 and 46–60 of the '604 patent. *See* IPR2013-00563. Microsoft indicates that claims 1, 2, 16, 17, 31, 32, 46, and 47 of the '604 patent are presently asserted against Microsoft in *Enfish LLC v. Microsoft Corporation, et al.*, Case No. CV12-7360 MRP (MRWx) in the Central District of California. Pet. 2. Microsoft further contends that a final judgment against Enfish has been entered in the California case. Ex. 1071.

Microsoft also has filed three petitions challenging the patentability of claims 1–60 of U.S. Patent 6,163,775 ("the '775 patent").[2] *See* IPR2013-00559, IPR2013-00560; IPR2013-00561.

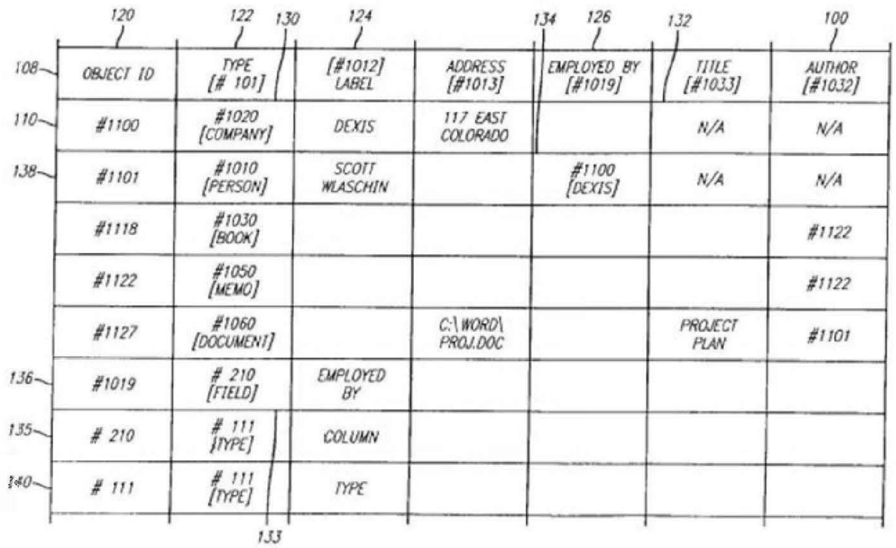
C. The '604 Patent

The '604 patent (Ex. 1001), titled "Method and Apparatus for Improved Information Storage and Retrieval," generally relates to a system and method for data storage, manipulation, and retrieval in a self-referential logical table of a database. This table is a logical structure, not a physical structure, stored contiguously in the memory. *Id.* at 6:29–35

Figure 3 of the '604 patent, reproduced below, illustrates a table structure of a logical table. *Id.* at 3:30–31.

---

[2] The '604 patent and the '775 patent both issued from continuations of application No. 08/383,752, filed March 28, 1995, now U.S. Patent No. 5,729,730. Pet. 6.



108	OBJECT ID	TYPE [# 101]	[#1012] LABEL	ADDRESS [#1013]	EMPLOYED BY [#1019]	TITLE [#1033]	AUTHOR [#1032]
110	#1100	#1020 [COMPANY]	DEXIS	117 EAST COLORADO		N/A	N/A
138	#1101	#1010 [PERSON]	SCOTT WLASCHIN		#1100 [DEXIS]	N/A	N/A
	#1118	#1030 [BOOK]					#1122
	#1122	#1050 [MEMO]					#1122
	#1127	#1060 [DOCUMENT]		C:\WORD\PROJ.DOC		PROJECT PLAN	#1101
136	#1019	# 210 [FIELD]	EMPLOYED BY				
135	# 210	# 111 [TYPE]	COLUMN				
140	# 111	# 111 [TYPE]	TYPE				

As depicted by Figure 3 of the ’604 patent, above, table 100 is defined by rows 108, 110, 138, 136, 135, and 140, and columns 120, 122, 124, and 126. Ex. 1001, Fig. 3. The intersection of a row and a column defines a cell in the table. *Id.* at 6:42–50. Each column corresponds to an attribute spanning various records. *Id.* at 6:38. An attribute is a single class description, such as an employer, denoted in column 126 of Figure 3 for example by the text “Employed By.” *Id.* at 1:63–65; 7:16–18.

Each row corresponds to a record spanning various attributes. Ex. 1001, 6:37–38. For example, row 110 corresponds to a company as shown in cell 130. *Id.* at 6:54–55.

The patent describes unique object identification numbers (OID) shown in Figure 3 beginning with the “#” sign.[3] The ’604 patent explains that each row is assigned a unique object identification number (OID) stored

[3] The “#” sign appears to have no convention aside from differentiating in the drawings the OID’s from other integers, e.g. reference numbers, entered data etc.

in column 120, (1100, 1101, 1118, etc.,) and each column is similarly assigned a unique OID stored in row 108 ([101], [1012], [1013], etc.). Ex. 1001, 6:42–50. According to the '604 patent it is desirable to have an associated *column definition* stored in the same table, as a row, to provide for better searching and synchronization between the columns and record data. *Id.* at Abst. A column definition determines the specific properties of a column. *Id.* at 7:11–14. For example, column 126 is defined as having a label "Employed By" (from the cell at column 124, row 136). The "Employed By" column contains employer information and OID [1019], and is defined by the record at corresponding row OID 1019, where column OID [1019] acts as a pointer to the record, thereby making the table self-referential. *Id.* at 7:16–19.

According to the '604 patent, the importance of the unique OID's is that they act as direct pointers to other records, which eliminates the necessity for a query to search through unrelated columns, rows, and cells for additional related data. Ex. 1001, 8:51–54.

D. Illustrative claims

Of the challenged claims, the independent claims are 1, 11, 12, 31, 41, and 42. Each of claims 2–10 depends, directly or indirectly, from claim 1. Each of claims 13–15 depends, directly or indirectly, from claim 12. Each of claims 32–41 depends, directly or indirectly, from claim 31. Each of claims 43–45 depends, directly or indirectly, from claim 12. Claims 1 and 31 illustrate the claimed subject matter and are reproduced below:

> 1. A data storage and retrieval system for a computer memory, comprising:
> means for configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row having an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column having an OID to identify each said logical column; and wherein

at least one of said logical rows has an OID equal to the OID of a corresponding one of said logical columns, and at least one of said logical rows includes logical column information defining each of said logical columns.

31. A method for storing and retrieving data in a computer memory, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said logical rows has an OID equal to the OID of a corresponding one of said logical columns, and at least one of said logical rows includes logical column information defining each of said logical columns.

E. The Prior Art References Supporting Alleged Unpatentability

Microsoft relies upon the following prior art references:

Chang et al., EP Publication No. 0 336 580 A2 (pub. Oct. 11, 1989) ("Chang," Ex. 1003).

Dickerson et al., EP Publication No. 0 394 019 A2 (pub. Apr. 18, 1990) ("Dickerson," Ex. 1003).

Crus et al., U.S. Patent No. 5,133,068 (issued Jul. 21, 1992) ("Crus '068," Ex. 1010).

Smith et al., U.S. Patent No. 5,181,162 (issued Jan. 19, 1993) ("Smith '162," Ex. 1008).

Goldberg et al., U.S. Patent No. 5,201,046 (issued Apr. 6, 1993) ("Goldberg '046," Ex. 1011).

Horn et al., U.S. Patent No. 5,226,158 (issued Jul. 6, 1993) ("Horn'158," Ex. 1005).

Jenness, U.S. Patent No. 5,463,774 (issued Oct. 31, 1995) ("Jenness '774," Ex. 1013).

Anderson et al., U.S. Patent No. 5,463,724 (issued Oct. 31, 1995) ("Anderson '724," Ex. 1012).

Covey, U.S. Patent No. 5,745,755 (issued Apr. 28, 1998) ("Covey '755," Ex. 1014).

KENNETH WEBB AND LORI LAFRENIERE, ORACLE DISTRUBUTED SYSTEMS-A C PROGRAMMER'S DEVELOPMENT GUIDE (1991) ("Webb," Ex. 1007).

F. The Pending Grounds of Unpatentability

The following chart summarizes Microsoft's patentablilty challenges:

Reference(s)	Basis	Claims challenged
Chang (Ex. 1003)	§ 102	1, 2, 31, and 32
Chang and Horn (Ex. 1005)	§ 103	12, 13, 42, and 43
Chang and Webb (Ex. 1007)	§ 103	11 and 41
Chang and Smith '162 (Ex. 1008)	§ 103	3, 8, 33, and 38
Chang and Dickerson	§ 103	4 and 34

(Ex. 1009)		
Chang, Dickerson and Crus (Ex. 1010)	§ 103	5 and 35
Chang and Goldberg (Ex. 1011)	§ 103	6 and 36
Chang and Anderson (Ex.1012)	§ 103	7 and 37
Chang, Smith '162, and Jenness (Ex. 1013)	§ 103	9 and 39
Chang and Covey (Ex.1014)	§ 103	10 and 40
Chang, Horn, and Smith '162	§ 103	14 and 44
Chang, Horn, Smith '162, and Jenness	§ 103	15 and 45

II. CLAIM CONSTRUCTION

A. Legal Standard

In an *inter partes* review, claim terms in an unexpired patent are interpreted according to their broadest reasonable construction in light of the specification of the patent in which they appear. 37 C.F.R. § 42.100(b); *see also In re Cuozzo Speed Techs., LLC.*, No. 14-01301, slip op. at 16, 19 (Fed. Cir. Feb. 4, 2015) ("Congress implicitly adopted the broadest reasonable interpretation standard in enacting the AIA," and "the standard was properly adopted by PTO regulation."). Claim terms are given their ordinary and customary meaning as would be understood by a person of ordinary skill in the art at the time of the invention and in the context of the entire patent disclosure. *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). If the specification "reveal[s] a special definition given to a claim term by the patentee that differs from the meaning it would otherwise

possess[,] . . . the inventor's lexicography governs." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316 (Fed. Cir. 2005) (en banc) (citing *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002)). Also, we must be careful not to read a particular embodiment appearing in the written description into the claim, if the claim language is broader than the embodiment. *See In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993) ("[L]imitations are not to be read into the claims from the specification."). We apply this standard to the claims of the '604 patent.

B. Overview of the Parties' Positions

Microsoft contends that this case involves a computer-implemented invention. Pet. 7. Microsoft also contends that the challenged independent claims invoke means-plus-function claiming, but the '604 patent fails to disclose an algorithm, which is an issue under § 112, sixth paragraph, that cannot be addressed in this proceeding. Pet. 14–15. In the Decision to Institute, we invited Enfish to direct us to the specific portions of the specification that clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. Inst. Dec. 11. For the reasons discussed below, we determine that Enfish in its response does not identify sufficient corresponding structure, as required under 35 U.S.C. § 112, sixth paragraph, for the "means for configuring said memory according to a logical table," recited in claims 1, 11, and 12.

Regarding terms in the method claims, in the Decision to Institute, we provided constructions for "logical table" and "object identification number," which are shown in the table below. Inst. Dec. 12–13. We also determined that no express construction is needed for the following terms:

"memory," "logical row," and "logical column." Inst. Dec. 15.

Claim Term or Phrase	Construction in the Decision to Institute
"logical table"	"[W]e construe the term "table" to mean: 'a structure of a database comprising rows and columns.'" Inst. Dec. 12. "We determine no express construction of 'logical' is needed for this decision." *Id.*
"object identification number"	"[W]e construe 'object identification number' in light of the specification to mean: 'an array of bits that define an object.'" *Id.* at 13.

Enfish contends that our construction for "object identification number" is incomplete. PO Resp. 12. We evaluate Enfish's contention below. We discern no reason, based on the complete record now before us, to change our constructions in the Decision to Institute of "logical table" and "object identification number."

C. Analysis of the Parties' Claim Construction Positions

1. Means for configuring said memory according to a logical table

Independent claims 1, 11, and 12 include the limitation, "means for configuring said memory according to a logical table." In the Decision to Institute, we agreed with Microsoft that under the broadest reasonable interpretation, the function for the means for configuring is "configuring memory according to a logical table." Inst. Dec. 11. Additionally, we considered the corresponding structure for the recited function as including a general purpose computer. *Id.* Enfish does not challenge persuasively either of these determinations; however, Enfish identifies portions of the specification that Enfish contends provide algorithmic support for the recited function. PO Resp. 17–18. In particular, Enfish contends that the '604

patent discloses a four-step algorithm that is linked to the recited function of configuring memory according to a logical table. *Id.*

"[T]he corresponding structure for a § 112 ¶ 6 claim for a computer-implemented function is the algorithm disclosed in the specification." *Aristocrat Techs. Austl. Party Ltd. vs. Int'l Game Tech.*, 521 F.3d 1328, 1333 (Fed. Cir. 2008) (quoting *Harris Corp. v. Ericsson Inc.*, 417 F.3d 1241, 1249 (Fed. Cir. 2005)). Additionally, specific portions of the specification must clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. *See Medical Inst. and Diag. Corp. v. Elektra AB*, 344 F.3d 1205, 1211 (Fed. Cir. 2003). For the reasons set forth below, we conclude that the four steps and other '604 patent specification portions identified by Enfish do not describe an algorithm for the recited function of "configuring memory according to a logical table." We further conclude that Enfish has not clearly linked or associated the recited function to the four steps or the other portions of the '604 patent specification that Enfish identifies.

For algorithmic support, Enfish identifies disparate excerpts of the '604 patent specification, which do not link or associate clearly a computer program or algorithm to the function corresponding to the claimed means for configuring said memory according to a logical table. Enfish also relies on Dr. Jagadish's Declaration, which as discussed below also does not show that the '604 patent specification provides an algorithm and does not link clearly any algorithm to the recited function. PO Resp. 17–18 (citing Ex. 2007 ¶¶ 69–74).

For example, the first step, "create, in computer memory, a logical table" (PO Resp. 17) appears to be similar to the recited function of

configuring memory according to a logical table, but the first step is not found in the '604 patent specification. Additionally, none of the portions of the '604 patent specification that Enfish cites for this first step provide an algorithm or computer program for performing the recited function of "configuring memory according to a logical table" or clearly link or associate any algorithm or program to this recited function. PO Resp. 17; *see also* Ex. 2007 ¶ 70 (citing Ex. 1001, 2:53–57, 6:35–38).[4] These portions describe that an already-formed table has rows and columns, without describing how memory is configured to create a logical table having rows and columns. Ex. 1001, 2:53–57. One portion states that memories "need not store" the table contiguously, but fails to describe an algorithm or computer program for configuring memory such that a logical table is not stored contiguously. *Id.* at 6:30–38.

Enfish's three remaining steps and the other '604 patent specification portions identified by Enfish fail to remedy these deficiencies. The second step is: "[a]ssign each row and column an object identification number (OID) that, when stored as data, can act as a pointer to the associated row or column." PO Resp. 17. One of the portions of the '604 patent specification cited by Enfish for the second step indicates "the system must generate a unique OID when columns and rows are formed." Ex. 2007 ¶ 71 (citing Ex. 1001, 8:6–7). The remainder of the identified portions relate to assigning an OID or indicate that an OID "may be used" as a pointer, without describing an algorithm for forming columns and rows of a table or showing how

[4] Enfish's citations in its Patent Owner Response appear to refer to column and line numbers of the '775 patent, rather than the '604 patent. Consequently, we use Enfish's citations to the '604 patent in Dr. Jagadish's Declaration.

assigning an OID relates to steps for forming columns and rows. *Id.* (citing Ex. 1001, Abstract, 2:54–55, 6:42–49, 6:60–61, 8:10–47, Figs. 3, 4). Additionally, the identified portions discuss assigning a numeric value to an OID in the form of a bit array, but fail to describe how to configure memory such that an OID may be used as a pointer. *Id.*

The third step (PO Resp. 18) and the portions of the '604 patent specification cited by Patent Owner for the third step (Ex. 2007 ¶ 72 (citing Ex. 1001, Abstract, 2:60–63, 6:39–40, 7:16–22, Fig. 3)) indicate that a table has a row that corresponds to columns, such as a header row, which is a generic feature of an already-formed table. These '604 patent specification portions identified by Enfish (*id.*), however, do not describe how to form a table with this feature or link or associate this feature to the recited function.

The fourth step, i.e., storing and accessing data in cells (PO Resp. 18) is performed in an already-configured table and, therefore, occurs after the recited function of configuring a table has occurred. Nonetheless, the '604 patent specification portions identified by Enfish (Ex. 2007 ¶ 73 (citing Ex. 1001, Abstract, 2:57–60, 6:40–41, 6:53–54, 7:1–2, 7:14–15, 11:47–59, Fig. 10)) suffer from the same deficiencies noted above.

We conclude that the four steps and other '604 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table. The portions of the '604 patent specification identified by Enfish describe an already-formed table having generic characteristics, such as columns, rows, identifiers, and a header row. *See e.g.*, Ex. 1001, Fig. 3. The description, however, does not disclose any algorithm or computer program for forming this table. Additionally, the description identified by Enfish is not linked or associated

clearly with the recited function of "configuring memory according to a logical table."

Enfish also relies on Dr. Jagadish's further statement that "[i]t is also my opinion that one of ordinary skill in the art would understand how to implement those algorithm steps using techniques and resources that were available at the time the '604 Patent was filed." Ex. 2007 ¶ 69. Enfish, however, cannot rely on the knowledge of one skilled in the art to fill in gaps. *See Function Media, LLC v. Google Inc.*, 708 F.3d 1310, 1319 (Fed. Cir. 2013).

Because we conclude that the four steps and other '604 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table, we terminate this proceeding with respect to the claims that recite this means-plus-function limitation. As explained in *BlackBerry Corporation v. Mobile Media Ideas, LLC*, Case IPR2013-00036 (PTAB Mar. 7, 2014) (Paper 65), the specification must disclose enough of a specific algorithm to provide the necessary structure under § 112, sixth paragraph. In the circumstance when the specification of the challenged patent lacks sufficient disclosure of structure under 35 U.S.C. § 112, sixth paragraph, the scope of the claims cannot be determined without speculation and, consequently, the differences between the claimed invention and the prior art cannot be ascertained. *Id.* For the reasons given, we determine that independent claims 1, 11, and 12 are not amenable to construction and, thus, we terminate this proceeding with respect to these claims under 37 C.F.R. § 42.72. Because claims 2–10 and 13–15 depend, directly or indirectly, from either claim 1 or claim 12, we also terminate this proceeding with respect to these claims under 37

C.F.R. § 42.72.

2. Means for searching said table for said pointer

Claim 41 is a method claim, and besides the step of "configuring said memory" includes a means-plus-function limitation, "means for searching said table for said pointer." Enfish contends that the specification and drawings of the '604 patent disclose an algorithm satisfying the necessity for corresponding structure under § 112 ¶ 6 for "a means for searching." PO Resp. 18–19. Enfish points to column 9 and Figures 5 and 6 of the '604 patent asserted as disclosing an algorithm for "searching" the table for the pointer. *Id.* Enfish contends that the specification and drawings describe:

> (1) Receiving a keyword and a column representative of an object that the OID pointer references. Ex. 1001, 9:7–10.
>
> (2) Retrieving the search path for the column to be searched for the keyword. Ex. 1001, 9:10–19.
>
> (3) Receiving search path information for the column to be searched for the keyword. Ex. 1001, 9:20-26.
>
> (4) Determining if more than one OID pointer referencing an object with the keyword is found and, if only one, return the OID pointer. Ex. 1001, 9:36–44.

Id. We find Enfish's evidence here persuasive in that the necessary structure in the form of an algorithm is set forth in the specification and drawings of the '604 patent such that one of ordinary skill in the art would have understood the structure, or acts, that perform the required "searching" function. *See Atmel Corp. v. Info. Storage Devices, Inc.*, 198 F.3d 1374, 1382 (Fed. Cir. 1999) (to satisfy 35 U.S.C. 112 "the corresponding structure(s) of a means-plus-function limitation must be disclosed in the

written description in such a manner that one skilled in the art will know and understand what structure corresponds to the means limitation."

Reviewing the specification, the first step in the algorithm is initiated by a user entering text "through the keyboard or mouse for a particular column that the user wishes to search." Ex. 1001, 9:1–2, Fig. 6. The database system thus, (1), receives the keyword for the desired column. For the second step (2), "the system retrieves the search path for the column to be searched." *Id.* at 9:3–4. This second step is further shown in block 152 in Figure 6 as "Fetch 'search path' and related options from associated column definition." Then, for step (3) as shown in block 154 of Figure 6, "the routine searches for a record that has an entry in the label column 124 of FIG. 2 that is the same as the text being searched for, and is of the same class." *Id.* at 9:14–17. Finally, at step (4) the routine determines if it has found at least one matching entry, and can thus return the respective OID for the object at block 168, as shown in Figure 6. *Id.* at 9:36–39.

In view of this description, we are persuaded that one of skill in the art would have been able to understand and implement the specific "searching" procedure contemplated in the '604 patent. The steps for this algorithm are articulated as a coherent "searching" procedure for obtaining an OID corresponding to a text entry and set forth in column 9, lines 1–36. Indeed, this searching procedure is shown and described with respect to Figure 5, and the flow chart in Figure 6 diagrammatically explaining the same procedure, which is linked to the claimed "means for searching." *See Atmel,* 198 F.3d at 1382 ("All one needs to do…is to recite some structure corresponding to the means in the specification…so that one can readily ascertain what the claim means and comply with the particularity

requirement of Para. 2.").

Accordingly, we agree with Enfish and identify the corresponding structure for performing the function of the "means for searching" in claim 41, namely, steps (1)–(4) as discussed above, which is a program, or routine, run on a specially programmed general computing device.

3. Other Means-Plus-Function Terms

Because we determine that this proceeding should be terminated as to claims 1–15 for the reasons discussed above, we need not construe other means-plus-function terms appearing in those claims for the purposes of this Decision.

4. Object Identification Number (OID)

Independent claims 31, 41, and 42 recite "object identification number (OID)" and "OID." In the Decision to Institute, we construed "object identification number" in light of the specification to mean: "an array of bits that define an object." Inst. Dec. 13 (citing Ex. 1001, 3:33–35, 8:35–37).

Enfish "agrees in principle" with our construction, but contends that the construction is incomplete. PO Resp. 12. Enfish contends that "OID" should be defined further as "a unique, immutable, and system-generated value that identifies an object." *Id.* at 12–13. Microsoft contends that Enfish seeks to read into the claims extraneous limitations that are unsupported by either intrinsic or extrinsic evidence. Pet. Reply 2. Microsoft additionally contends that Enfish's proposed construction does not provide an appropriate context for the additional limitations. *See e.g.*, Pet. Reply 3.

At the heart of Enfish's contention is an assertion that an OID will not function properly if it is not unique, immutable, and system generated. PO

Resp. 13–16. For example, Enfish, relying on Dr. Jagadish, contends that an OID must be a unique value because "if an OID were not unique the database would be non-functional because objects could not be reliably retrieved." PO Resp. 13 (citing Ex. 2007 ¶¶ 46–54). Dr. Jagadish cites to a portion of the specification stating that the OID is used for "exact retrieval" and that without uniqueness, retrieval using an OID will result in more than one object. Ex. 2007 ¶ 51 (citing Ex. 1001, 1:60–64).

Microsoft contends that Enfish's proposed construction should not be adopted. Pet. Reply 2–7. For example, Microsoft contends that Enfish's "unique" OID requirement would conflict with the patent, which describes a row in a table with an OID that has the same value as the OID of a column in the same table. Pet. Reply 3 (citing Ex. 1001, Fig. 3). Microsoft also contends that "immutable" should not be imported into the construction of OID because the term is absent from the specification, claims, and file history and the specification is devoid of disclosure relating to immutability of an OID. Pet. Reply 4–5. As further support, Dr. Hosking states that OID may change in certain circumstances and still retrieve information reliably. Ex. 1046 ¶¶ 23–25.

We determine that adopting Enfish's proposed construction without further context would create ambiguity. Additionally, limitations are not to be read into the claims from embodiments in the specification. *See In re Van Geuns*, 988 F.2d 1181 at 1184. Furthermore, we agree with Microsoft that Enfish relies on insufficient extrinsic evidence for support, for example, that an OID is "immutable." We also note that Enfish's contentions are not commensurate with the scope of the claims because each of the independent claims recites that an OID is included "to identify" each logical row and

each logical column, but does not recite using an OID to retrieve data from memory.

We, therefore, determine that the construction of "OID" in the Decision to Institute should not be changed.

III. ANALYSIS

A. Anticipation by Chang

We have reviewed Microsoft's anticipation argument and supporting evidence, including Chang's disclosure and the detailed explanation appearing on pages 21–38 of the Petition. We are persuaded that Microsoft's analysis demonstrates that each limitation of claim 31 is present in Chang. Despite the counter-arguments in Enfish's Patent Owner Response, and the evidence cited therein, which we also have considered, Microsoft has shown, by a preponderance of the evidence, that claim 31 is unpatentable under 35 U.S.C. § 102(b) as anticipated by Chang. *See* 35 U.S.C. § 316(e).

1. Chang

Chang discloses a relational database including EMPLOYEE TABLE 10 ("EMP") in Figure 1 which includes a number of records, each record (row) containing specific data relating to a certain employee. Each record in EMPLOYEE TABLE 10 includes the *employee number*, name, salary, and department number aligned in columns 1–4 respectively. Ex. 1003, 5:26–33. The relational database also includes other tables, i.e. system catalogs, as shown in Figures 2–4. The system catalogs contain further information about the tables in the database, for example SYS.TABLES 12 in Figure 2 contains a record for EMPLOYEE TABLE 10 including the table name, and

the creator in columns 1 and 2. Chang states that SYS.TABLES 12 may also contain information about a different table, such as an employee location table (not shown) including the *employee number* and location. Chang explains that:

> Due to common elements in these various tables relational information may be derived by using this commonality. For example, because the *employee number* is common to the aforementioned employee table and site location table, this piece of information may be used to relate a particular employee's name from the first table to the location where the employee works derived from the second table.

Id. at 5, 45–53 (emphasis added).

Chang also discloses how to store definitional information relating to column attributes in a single record (row) of a table so that "[a]ccessing the row corresponding to a particular object returns a description of all of the attributes of the object's component objects, as well as information describing the object itself." *Id.* at 3:57–4:3. More specifically, Chang explains that the table shown in Figure 3, as reproduced below, SYS.COLUMNS catalog 14, contains records relating to the database structure, i.e., database objects and component objects, of the EMPLOYEE TABLE 10.

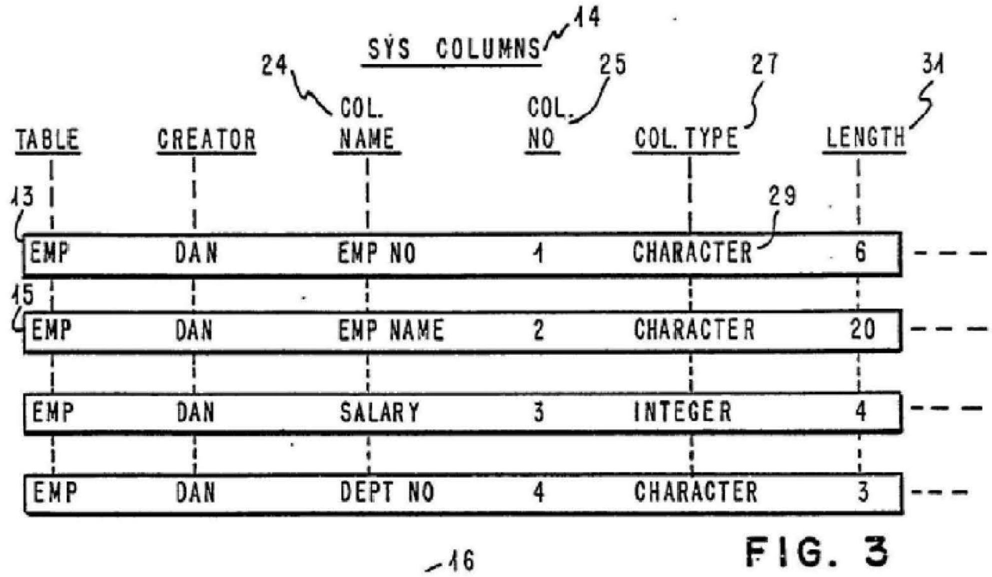


FIG. 3

Figure 3, above, reveals that SYS.COLUMNS catalog 14 contains specific information about the attributes of each of the columns in the EMPLOYEE TABLE 10. *Id.* at 6:25–39. Chang describes:

> each record in the SYS.COLUMNS catalog14 shown therein contains a number of fields describing attributes of a correlative one of the columns in the employee table 10…for example, EMP in the “Table” column as the first field thereof indicates that the data to the right is further information or attributes about a column appearing in the employee table.

Id. at 6:40–54. Moreover, still referring to Chang’s Figure 3, each record (row) in SYS.COLUMNS catalog 14 includes a field (“COL.NO” 25) having the column number relating to the corresponding column in the EMPLOYEE TABLE 10, which is defined by that particular row. For instance, observing the SYS.COLUMNS catalog 14 in Figure 3, it is clear the value “1” in COL.NO 25 correlates this record with column 1 in the EMPLOYEE TABLE 10 shown in Figure 1. *Id.* at 6:50–7:10.

2. Claim 31

We have reviewed Microsoft's anticipation argument and supporting evidence which relates persuasively each element of claim 31 to the disclosure of Chang. Pet. 22–29.

Microsoft asserts that Chang discloses a relational database and that it is well known that databases are used to store and retrieve data. *Id.* at 22 (citing Ex. 1003, 1:3–4, Abst., Ex. 1022 ¶¶ 26, 158–159). Microsoft contends that Chang describes implementing the database on a computer system having memory, a central processing unit, and a display. *Id.* (citing Ex. 1003, Fig. 11). Microsoft further asserts that Chang discloses a method for configuring the memory according to a logical table that organizes data into rows and columns. *Id.* at 23–24 (citing Ex. 1003, 13:57–14:10, Figs. 2–5, 11, Ex. 1022 ¶¶ 173–175, 182–184). The OID's identifying each row, Microsoft contends, are disclosed at least by Chang's specific discussion stating for SYS.COLUMNS table, "that a row will have a 'record i.d.' associated with it." Pet. 26 (citing Ex.1003 at 7:51–8:3). Also, Microsoft asserts that SYS.COLUMNS includes an OID for each column, "because SYS.COLUMNS stores the definition of all columns in the tables. . . including the columns of SYS.COLUMNS itself . . . Chang teaches that the columns of the SYS.COLUMNS table shown in Figure 3 will each have a number to identify them." Pet. 28–29 (citing Ex. 1003, 5:35–38, 5:54–55, 6:25–39, 6:58–7:3, 8:45–50). Microsoft's arguments here relate to portions [A], [B], [C], and [D] of claim 31 for example, reproduced below. We are persuaded that Chang discloses these initial limitations of claim 31 because, as Microsoft's evidence shows, Chang describes a relational database having objects such as tables, and columns for storing data in a determinable way.

Ex. 1003, 1:1–14. Chang further explains how various data entry, search, and manipulation is carried out with respect to such objects. *Id.* at 27–37, Also, Chang discloses, in accordance with our claim construction, that each database table includes column, and row, identifiers, such as record i.d., column numbers, or columns names, comporting with the proper interpretation of an OID, as "an array of bits that define" a column. *Id.* at 5:41–45.

Turning to the remaining limitations in claim 31, referenced as [E] and [E2], Enfish makes three specific arguments regarding why Chang does not disclose all the limitations of independent claim 31. PO Resp. 20. For purposes of clarity, we reproduce below claim 31, including the additional reference elements ([E1] and [E2]) added by Enfish.

> 31. [A] A method for storing and retrieving data in a computer memory, comprising the steps of:
>
> [B] configuring said memory according to a logical table, said logical table including:
>
> [C] a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;
>
> [D] a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein
>
> [E1] at least one of said logical rows has an OID equal to the OID of a corresponding one of said logical columns, and [E2] at least one of said logical rows includes logical column information defining each of said logical columns.

Enfish asserts first that no one, *single table* in Chang includes all the limitations of these independent claims. PO Resp. 20. Second, Enfish contends that neither of Chang's two tables (SYS.TABLES,

SYS.COLUMNS), discloses the row OID limitation [C], and third, neither table discloses the limitation [E1] requiring a row OID to be equal to a column OID. *Id.*

Enfish's contentions are based on its proposed construction of OID, "a unique, immutable, and system-generated value that identifies an object," which we decline to adopt for the reasons given above. *See* PO Resp. 12–13. In addition, we disagree with Enfish's first assertion. While we are persuaded that claim 31 requires the elements to be found in a single table, Microsoft has provided persuasive evidence that Chang's SYS.COLUMNS table discloses each limitation of the claimed "logical table."

To anticipate a patent claim under 35 U.S.C. § 102, "a single prior art reference must expressly or inherently disclose each claim limitation." *Finisar Corp. v. DirecTV Group, Inc.*, 523 F.3d 1323, 1334 (Fed. Cir. 2008). Microsoft's evidence is persuasive that Chang's SYS.COLUMNS table, at least inherently if not expressly, discloses a record/row that defines each of the columns that appear in the SYS.COLUMNS table itself. Pet. 33 (citing Ex. 1022 ¶¶ 169–175). SYS.COLUMNS, although shown by way of example in Figure 3 of Chang for EMPLOYEE TABLE ("EMP"), will, in the Table field also have a correlating SYS.COLUMNS row, defining the various columns of the SYS.COLUMNS table. *See* Pet. 33, Ex. 1003 at 6:49–7:19. In light of our interpretation of OID set forth above in II.C.3, we are persuaded by Microsoft's argument for element [E1] that "[a] row holding a column definition will of course hold an OID that is the same as the OID of the defined column." Pet. 33.

Enfish's second position is that the OID's identified by Microsoft do not satisfy the "each said logical row including an OID" part of claim

limitation [C] "because there is no cell in any row of Annotated FIG. 3 that contains this supposed OID." PO Resp. 27. We are not convinced by Enfish's argument. Enfish did not present any claim construction for the term "logical row including." The limitation in claim 31 reads in context "each said logical row including an object identification number (OID)," not "*a cell of* each said record including an object identification number (OID)." Indeed, the evidence asserted by Enfish, specifically Dr. Hosking's deposition testimony, indicates that an OID *can* be stored in a cell, not that it *must* be stored in a cell, in order to identify the row. *Id.* at 26–27 (citing Ex. 2003, 183:12–17. Enfish's argument is not commensurate in scope with the claims. Accordingly, we are not apprised by Enfish of any persuasive evidence from the specification of the '604 patent, or otherwise, that the claims require that an OID *must* be stored in a cell of either a row or column.

In its third argument, Enfish maintains that the [E1] limitation, "at least one of said logical rows has an OID equal to the OID of a corresponding one of said logical columns," is not meet by Chang because none of the OID values considered by Microsoft (record i.d., column number, column name, a combination of the table name, creator name, and/or column name or number) "satisfy all three of the requirements of OIDs—uniqueness, immutability and system generated." PO Resp. 30. This argument is not persuasive because it addresses the limitation [E1] in terms of the wrong claim construction.

We determine that the values, alone and in combination, identified by Microsoft as OID's satisfy our construction for an OID as "an array of bits that define." Under the proper claim construction, each row in SYS.COLUMNS, shown in Figure 3 of Chang, includes for example both

the column name (column 3), as well as column number (column 4) as an OID. *See* Pet. 33, annotated Fig. 3. Therefore, where at least the column name and column number values can function as OID's, and with such OID values for the columns, being included in the row defining a particular column of SYS.COLUMNS table, the single SYS.COLUMNS table thus satisfies the [E1] limitation "wherein at least one of said logical rows has an OID equal to the OID of a corresponding one of said logical columns."

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 31 of the '604 patent is anticipated by Chang.

3. Claim 32

Enfish further argues that logical column information defining "one of said logical columns to contain information for enabling determination of OID's from text entry," as called for in claim 32, is not found in a single table, but is only found in SYS.INDEXES catalog of Figure 4, which is not a part of SYS.TABLES (Figure 2) or SYS.COLUMNS (Figure 3). PO Resp. 42–43. Microsoft does not address this argument in their Reply.

As discussed above, we are persuaded that claims 31 and 32 require all the claim elements following "said logical table including" to be found in the same logical table. The Federal Circuit has repeatedly emphasized that the indefinite article "a" carries the meaning of "one or more" in open-ended claims containing the transitional phrase "comprising." *Baldwin Graphic Sys., Inc. v. Siebert, Inc.*, 512 F.3d 1338, 1342 (Fed. Cir. 2008). Thus, claim 31 recites "a logical table," and while the claim is not restricted to a database with one table, we determine that the recitations in claim 31 and 32 following "said logical table including" must be found in a single table.

Claim 32 further limits the "logical column information" recited as an

element of the logical table in claim 31. Microsoft does not identify in the specification of the ’604 patent, nor point us to any evidence or embodiment, indicating that the elements of Enfish’s claimed “logical table” were intended to be in separate, different tables. *See* Ex. 1001, Abst., 1:41–60, 3:7–21. The specification of the ’604 patent consistently refers to table 100 in the context of a single table, including, in an embodiment used in a word processing application. *Id.* at 16:65–67, Figs 3, 9, 10, 13. (“The database of the present invention includes a novel Structured Word Processor that may be used in conjunction with the table 100.”)

We conclude that the evidence provided by Microsoft with respect to the packed description in SYS.TABLES storing column information, including primary key column definition information from SYS.COLUMNS table is not persuasive because it does not explain how OID determination by text searching as recited in claim 32 would be conducted on such stored information and definitions in these particular tables, or even that SYS.TABLES or SYS.COLUMNS are indexed to provide such a search function. Pet. 34–35. Dr. Hosking describes that it is the SYS.INDEX of Figure 4 that is necessary for Chang to determine what “columns are amenable to being searched.” Ex. 1022 ¶ 184. Enfish argues persuasively that the only table Chang discloses for index searching is SYS.INDEX shown in Figure 4. PO Resp. 42–43 (citing Ex. 1003, 7:27–29, Ex. 2003 at 190–193:7–15).

For the foregoing reasons, we determine that Microsoft has not shown, by a preponderance of the evidence, that claim 32 of the ’604 patent is anticipated by Chang.

B. Obviousness over Chang and Horn

For the reasons given below, after consideration of Enfish's Patent Owner Response, and the evidence cited therein, we determine that Microsoft has shown, by a preponderance of the evidence, that claim 42 is unpatentable as obvious over the combination of Chang and Horn. We further determine that Microsoft has not shown by a preponderance of the evidence that claim 43 is unpatentable over Chang and Horn.

1. Horn

Horn discloses a method for maintaining the referential integrity of a relational database by reviewing, i.e. examining, prior to altering the database, any proposed changes to the database and determining if the proposed changes affect any known relationships in other data tables. Ex. 1005, Abst. Horn explains that in a referential database, it is important "to efficiently and rapidly ascertain whether or not an attempted modification to the structure of a relational database will allow consistent referential integrity." *Id.* at 4:48–51. Horn shows in Figure 2 an "employees" table that contains an employee number column "EE NO" where the employee number for a record object (e.g. employee Sharp) may also be found in a different "MANAGER EE" column indicating a manager relationship with respect to a different record object (i.e. employee Abel).

2. Claim 42

Enfish argues that Microsoft does not provide any explanation of how Horn's employee table at Figure 2 can be combined with Chang's system catalogs, i.e. SYS.TABLES and SYS.COLUMNS, which have no relational data from the employee table. PO Resp. 50. Enfish specifically argues "Petitioner merely proposed combining Horn with the Employee Table (FIG. 1) of Chang, and makes no mention of how Horn can be combined

with the system catalogs to disclose the single table recited in claims 12 and 42." *Id.*

Microsoft asserts that from a combination perspective, Horn represents prior art confirming Dr. Hosking's testimony that it was widely known in the art at the time of the invention of the subject matter of the '640 patent, as a matter of design choice, to have separate rows with some relationship in a table refer to one another. *Id.* at 38–39 (citing Ex. 1022 ¶¶ 269–270). Microsoft explained that "[a]n employee table is a mere example where this occurs, but a person of ordinary skill understood to create these relationships as necessary, and it would have been obvious to employ row-to-row pointers in the system tables as well as the employee table of Chang's Figure 1. *Id.* at 39 (citing Ex. 1022 ¶ 271). Dr. Hosking testified as follows:

> 271. The row-to-row pointers shown in Horn were a common tool available to database designers from well before the priority date of the '604 patent. Horn's use of that common tool is not treated as an innovation within Horn because, as a general matter, it was well-known to use row-to-row pointers whenever an association (relation) is proper between two elements in a table. The employee table in Horn is an example of a table with the row-to-row pointers.

Ex. 1002 ¶ 271. We are persuaded by Microsoft's argument because Horn's employee table example teaches that a cell in a row, such as record object employee Abel, directs, or points, to a different row, e.g., record object employee Sharp. Ex. 1002 ¶ 270. The predictable use of familiar prior art elements according to their established functions renders the recited invention obvious. *See KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 417 (2007).

Enfish argues that Horn's reference between associated rows is not a

"pointer" within the plain meaning of the term, as understood by those of ordinary skill in the art. PO Resp. 50. Enfish asserts that a "pointer" is understood in the field as "a variable that stores an address to a location where an object resides" and that a shared employee number is not a variable that stores an address. *Id.* Enfish's argument does not, however, explain why the stored employee number, in the "MANAGER EE" column for employee Able, is not an address, or is not storing an address, identifying record object employee Sharp, i.e. Able's manager.

We are not persuaded that Enfish has sufficiently rebutted Microsoft's explanation that one of ordinary skill in the art would understand the shared employee number as a pointer in the context of functionally relating one object to another, e.g. relating row object employee Sharp, to row object employee Able.

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 42 of the '604 patent is unpatentable as obvious over the combination of Chang in view of Horn.

3. *Claim 43*

Enfish incorporates by reference its arguments from the Preliminary Response, with respect to claims 7 and 37, which recite the same subject matter as claim 43. PO Resp. 51 (citing Prelim. Resp. 56–58). Enfish argues that Microsoft relied upon Anderson in claims 7 and 37 to teach the "pointers" that apparently Chang lacks, but, does not rely upon Anderson for claim 43. Enfish argues specifically that "[t]he Board should interpret Petitioner's statements in claims 7 and 37 as an admission that Chang does not disclose the 'plurality of pointers' recited in claims 13 and 43. *Id.* at 54.

Microsoft argues that the limitation of claim 43, "at least one of said

logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contains defined values," is met in part by Chang's disclosure of a cell in a row of SYS.TABLES containing the Packed Description. Pet. 42. Microsoft further states that Chang discloses logical pointer in "COL.NO" column in SYS.COLUMNS table pointing to the column's logical placement in a table. *Id.* at 43 (citing Ex. 1003, 7:1-3, Fig. 6).

We are not persuaded by Microsoft's position because the argument relies upon at least two different tables in Chang, SYS.TABLES and SYS.COLUMNS. This evidence fails to show the limitations in claim 43 as being within a single table as discussed above in section III.A.3. Claim 43 further limits the logical columns of "said logical table" as recited in claim 42, thus, the limitations of claim 43 must also be found within a single table.

We conclude that Microsoft has not shown by a preponderance of the evidence that claim 43 would have been obvious over Chang and Horn.

C. Obviousness over Chang and Webb

We have reviewed Microsoft's obviousness arguments and supporting evidence, including Chang and Webb's disclosure and the detailed explanation appearing on pages 40–42 and 46–48 of the Petition. We determine that Microsoft's explanation persuasively reads all elements of claim 41 onto the disclosure of Chang.

1. Webb

Webb describes a distributed application which runs on a local computer and accesses data stored on a central computer. Ex. 1007, 14. The central computer makes the same data accessible to multiple users and

application programs. *Id.* Webb describes itself as a "[g]*uide* . . . for C programmers who want to embed Structured Query Language (SQL) statements into their programs for access to local or remote (distributed) databases." *Id.* Webb also discloses table 3-5 including several rows, PRODUCT_ID 102 through 108, each row having pointer 101, that points to PARENT_PRODUCT_ID 101 which is a separate record/row. *See* Ex. 1007 at 47.

2. Claim 41

Enfish argues that Microsoft does not provide sufficient rationale for combining Chang and Webb. PO Resp. 51–52. Enfish specifically argues "[n]either the Petitioner nor Dr. Hosking, however, have even discussed such a combination, let alone demonstrated that such a combination is possible." *Id.* at 52 (citing Prelim. Resp, 53–56). We are not persuaded by this argument. In contrast to this statement, Microsoft explained that Webb provides a flexibility desired by relational database designers and relates to a relational database system and specific database structure, which includes SQL (Structured Query Language) statements, and discloses table 3-5 where at least one of the records/rows has a cell that contains a pointer to a different row." Pet 44–46 (citing Ex. 1007 at 47, Ex. 1022 ¶¶ 287–288.).

We are persuaded that Microsoft has shown a sufficiently articulated reason with rational underpinning to support obviousness. Pet. 47–48 (citing Ex. 1022 ¶¶ 289–297). Microsoft explained that one of ordinary skill in the art would have understood the SQL language and would have combined Webb's database structures with Chang's system in order to "consolidate[e] system catalog information to reduce access requirements within a relational database." *Id.* at 48. The predictable use of familiar prior art elements

according to their established functions renders the recited invention obvious. *See KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 417 (2007).

Microsoft explains how the initial limitations [A]–[D] of claim 41 are met by Chang for the same reasons as claim 31 discussed above in section III.C.2. Pet. 43. Microsoft additionally explains how the remaining elements of claim 41, listed below as [F1], [F2], and [F3] as referred to by Microsoft and Enfish, are met by Webb. Pet. 44–46 (citing Ex. 1007, xiii, 47).

> [F1] at least one of said plurality of logical rows contains a logical cell that contains a pointer to a different logical row and
>
> [F2] at least one of said plurality of logical rows includes information defining the type of a different logical row; and
>
> [F3] means for searching said table for said pointer.

Id. at 44. Microsoft contends that Webb discloses table 3-5 having several rows that point to the parent product ID of a different record/row. Pet. 47 (citing Ex. 1007, 47). Microsoft further argues "that rows with PRODUCT_ID of 102 through 108 are all of the same type, i.e. category, defined by the row for PRODUCT_ID 101, as indicated in the PARENT_PRODUCT_ID column. *See* Ex. 1007 at 47. Observing table 3-5 in Webb, the first column entry of the first record "Product ID" 101 is referenced by the following seven records at PARENT_PRODUCT_ID column. Further, the first record and column entry "Product ID" 101 defines each of the seven following records in table 3-5 as a component of the first record product "Professional Oracle." We find this evidence persuasive that

the value 101 in table 3-5 acts as a pointer, cross-referencing records to interrelate and identify the type or category of certain records. Chang discloses elements [F1] and [F2] of claim 41.

As discussed above in claim construction section II.C.2, claim 41 also includes the limitation "means for searching said table for said pointer." Microsoft explains how this means-plus-function limitation, and its corresponding structure, is met by Chang. Pet. 41–42, 46 (citing Ex. 1003, 10:34–11:4, Figs. 2, 4, 5, 6, 7, 11, Ex. 1022 ¶¶ 261–262).

> Chang discloses searching a relational database using a general purpose computer. *See* Ex. 1003 at Figure 11. Chang discloses which columns are amenable to being searched by disclosing that certain columns are indexed. *See* Ex. 1003 at Figure 4. This information is collected into the same packed descriptor, PD, as the other column defining information discussed with respect to claim 31. *See* Ex. 1003 at Figures 2, 6, 7, 10:34–11:4. Searching the index is meant to return a particular row or rows, from which the OIDs described for limitation 31[C] can be determined. *See* Ex. 1002, Hosking Decl. at ¶¶ 261-262. Chang stores the column information in the Sys.Columns table, which is also stored in the packed descriptor, PD, of Figure 2. *See* Ex. 1003 at Figures 2, 6, 7, 10:34–11:4. Accordingly, Chang discloses searching on a general purpose computer using an index. See Ex. 1002, Hosking Decl. at ¶¶ 261-262.

Pet. 41–42. Enfish does not specifically argue that Chang fails to meet this means-plus-function claim limitation, but relies upon reference to sections VII.A.1, and VII.A.1.b, of its Patent Owner Response that argue Chang fails to disclose the column and row OID limitations recited in the independent claims. *See* PO Resp. 51–52. As discussed above with respect to claim 31, we do not find Enfish's position persuasive. We are persuaded by Microsoft's explanation, above, that effectively reads the searching function

described in Chang, on the algorithm structure of "means for searching."

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 41 of the '604 patent is unpatentable as obvious over the combination of Chang in view of Webb.

D. Obviousness over Chang and Smith '162, or over Chang, Horn, and Smith '162, or over Chang, Smith '162, and Jenness, or over Chang, Horn, Smith '162, and Jenness

We have reviewed Microsoft's obviousness arguments and supporting evidence, including Chang and Smith '162's disclosure and the detailed explanation appearing on pages 46–48 of the Petition. For the reasons given below, we determine that Microsoft has not shown, by a preponderance of the evidence, that claims 33, 38, 39, 44, and 45 are unpatentable as obvious over the combination of Chang and Smith '162.

1. Smith '162

Smith '162 describes an object-oriented database system where various documents are organized and represented as collections of logically related documents, i.e. documents of a similar type, or "objects." Ex. 1008, Abst. These "objects" containing the document collections can be "generically referred to as 'folders,'" and the folders represented as "objects." *Id.* at 3:29–35. The underlying database management system has an organizational table structure with rows and columns, where columns can specify objects, including attributes of an object, representing a folder. *See Id.* at 9:66–68, 10:31–34, Fig. 1.

2. Claims 33 and 38

Enfish asserts that Microsoft fails to provide a sufficient rationale for combining Chang and Smith '162. Prelim. Resp. 38. Enfish contends that

Microsoft's only motivation for the combination is that "'Chang, like Smith '162, describes columns and rows as being represented by objects.'"Prelim. Resp. 38 (citing Pet. 47:10–15). Enfish argues that the bare assertion by Microsoft and Dr. Hoskings, that two references are within a similar field of endeavor, is insufficient to support a determination of obviousness. *Id.* (citing Ex.1022 ¶ 281).

We agree with Enfish that Microsoft has not met its burden to show that a person of ordinary skill in the art would have modified Chang based on the disclosure of Smith '162 resulting in the subject matter of the challenged claims. Microsoft asserts that "a person of ordinary skill in the art would have been motivated to combine Chang with Smith '162 because Chang, like Smith '162, describes columns and rows as being represented by objects and Chang describes including a "file i.d." describing a location in a column." Pet. 47 (citing Ex. 1008, 10:31–34, Ex. 1003, Abst., Fig. 3). Microsoft contends that Chang arguably does not explicitly disclose folders but a person of ordinary skill in the art would combine the folder objects of Smith '162 with Chang.. Pet. 47–48 (citing Ex. 1008, 8:12–13). Microsoft further contends that Smith '162 discloses folder objects having pointers that point to other objects, such as documents, within the same folder. Pet. 48 (citing Ex. 1008 8:19–20).

Our review of Chang reveals simply that a "file i.d.," represents a file location "from which additional information on the table may be derived." Ex. 1003, 6:5–10. Microsoft's statements indicating that both references describe elements of tables in a database does not provide sufficient explanation or evidence for combining these references. *See* Pet. 47–48.

A claim is unpatentable under 35 U.S.C. § 103(a) if the differences

between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. *See KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). A patent claim composed of several elements, however, is not proved obvious merely by demonstrating that each of its elements was known, independently, in the prior art. *Id.* at 418. In analyzing the obviousness of a combination of prior art elements, it can be important to identify a reason that would have prompted one of skill in the art to combine the elements in the way the claimed invention does. *Id.* Microsoft's argument does not explain why one of skill in the art would look to Smith '162 for a "folder type record." Microsoft has not articulated any reason, or presented rational evidentiary underpinnings, explaining why Chang's "file i.d." was, or should have been, understood as a "folder object" as disclosed by Smith '162, or why one of skill in the art at the time of invention of the subject matter of the '604 patent would have looked to combine Smith '162's "folder objects" with Chang's "file i.d."

We conclude that Microsoft has not shown by a preponderance of the evidence that claims 33 and 38 would have been obvious over Chang and Smith '162.

3. Claims 39, 44, and 45

For claim 39, Microsoft relies on the combination of Chang, Smith '162, and Jenness. Pet. 52–53. Microsoft contends that Jenness teaches a pointer to the folder allegedly disclosed by Smith '162. *Id.* (citing Ex. 1013 5:62-63). As explained above with regard to claims 33 and 38, we conclude that Microsoft has not articulated any reason, or presented rational

evidentiary underpinnings, explaining why Chang would have been combined with Smith '162. Thus, claim 39, which depends from claim 38, is not obvious for the same reason as claim 38.

To meet the limitations of claim 44, Microsoft relies upon the combination of Chang, Horn, and Smith '162, and states that "it would have been obvious to combine Chang and Smith '162, as described for claim [] 38, with the further teachings of Horn. Pet. 55. Chang and Smith '162, however, cannot be combined to meet claim 44 for the same reasons discussed above in regard to claim 38. Claim 45 depends from claim 44, thus for the same reasons discussed above, Chang and Smith '162 cannot be combined to meet claim 44. Accordingly, we conclude that Microsoft has not shown by a preponderance of the evidence that claims 39, 43, and 44 would have been obvious over Chang, Smith '162, and Jenness, and/or Horn.

E. Obviousness over Chang and Dickerson, or over Chang, Dickerson, and Crus

For the reasons given below, after consideration of Microsoft's Petition and Enfish's PO Response, and the evidence cited therein, Microsoft has not shown by a preponderance of the evidence that claims 34 and 35 are unpatentable as obvious over the combination of Chang and Dickerson, or Chang, Dickerson, and Crus.

1. Dickerson

Dickerson discloses a database system residing on a host, or central computer, and users can access the database from terminals connected to the central computer. Ex. 1009, 1:25–45, Fig. 1. Dickerson also teaches a contingency system having the same database at two different sites, with

both sites being accessible by users from other terminals, where both databases are updated to maintain the databases identical. *Id.* at 11:5–20. Dickerson states "[w]henever a resource such as a part of a database is accessed during a sync interval, a lock will be placed on both copies of that resource (one copy at each site) . . . the two databases will always remain identical for practical purposes as the locks on the data at the second site will not be released until its commit process is completed, so that no terminal will ever be able to obtain differing information from the two databases." *Id.* at 11:9–20.

2. Claim 34

Microsoft argues that Chang discloses all the limitations of base claim 31, but not the reciprocal synchronization of two logical columns recited in claim 34. Pet. 48. More particularly, Microsoft asserts that Dickerson discloses the limitation in claim 34 "wherein said attribute set information defines one of said logical columns to contain information for synchronizing two logical columns reciprocally." *Id.* Microsoft argues that Dickerson discloses "a replicated database" that teaches "reciprocal synchronization of two logical columns," where Dickerson discloses modification of a column in one database replicated in databases at other sites. *Id.* at 48–49 (citing Ex. 1009 at 10:56–11:28, Figure 8). In support of this ground, Microsoft relies on the Hosking Declaration (Ex. 1022 ¶¶ 225, 285–294), explaining how each limitation is disclosed in Chang and Dickerson. Pet. 49.

Enfish contends that Microsoft's arguments are based on "mirroring" that "occurs across separate databases as opposed to across two columns within the same table." PO Resp. 44. Enfish makes two specific arguments with respect to "mirroring" and the asserted combination. *Id.* at 44–45.

Enfish first contends that Petitioner has failed to explain, or provide evidence, why one of skill in the art would combine these references to "mirror" columns within the same logical table in the first place. *Id.* at 45. Enfish asserts that on the contrary, one of skill in the art would understand that "mirroring" columns across different databases defeats the purpose of a single system catalog such as SYS.TABLES and SYS.COLUMNS in Chang. *Id.* Also, Enfish contends that "mirroring" occurs across separate databases and is not the same function as "synchronizing two logical columns reciprocally" as recited in claim 34. *Id.*

Dickerson teaches replication of data across separate databases at different sites by synchronization:

> Whenever a resource such as a part of a database is accessed during a sync interval, a lock will be placed on both copies of that resource (one copy at each site). A communications function local to the transaction manager will participate in the commit process with that transaction manager so as to guarantee that any update information is transmitted out of that site by the time that commit takes place. The transmitted update information is used to update the database at the other site so as to maintain the two databases identical.

Ex. 1009 11:9–20. Dickerson states that mirroring "is an arrangement in which databases at different sites hold the same data, as in a contingency system, but with the databases at both sites being available for access by terminals." *Id.* at 10:58–11:4. The evidence in Dickerson relating to mirroring data in separate databases reveals a discrete sync interval permitting the transfer of the same data, in the same attribute set, between separate databases and separate tables. *Id.* at 5:56–6:5. This disclosure, however, does not describe how to synchronize two attribute sets in the same

table by a specific synchronizing attribute set, also in the same table, as called for in claim 34.

We give little weight to Dr. Hosking's testimony with respect to the limitations in claim 34 because he does not provide an explanation or evidence as to why Dickerson's synchronization of data in the same column across separate databases teaches reciprocal synchronization between columns in the same table. Dr. Hosking states that a person of skill in the art would be motivated to combine Chang and "the synchronization concept as taught in Dickerson in the case that similar elements were to be reciprocally synchronized." Ex. 1022 ¶ 225. The premise of this argument is the assumption that Dickerson's synchronization across separate databases is the same as synchronizing two logical columns reciprocally in the same database table. Dr. Hosking does not provide support for this assumption. This argument, therefore, does not sufficiently explain why mirroring data across separate databases leads to data synchronization between logical columns in the same table. Dr. Hosking's testimony does not explain why, or how, one of ordinary skill in the art would use the database management disclosed by Chang in cooperation with the linked database elements between separate databases to achieve reciprocal synchronization "to achieve such coordination of the data" between logical columns in the same table. *See id.* at ¶¶ 283–294.

We conclude that Microsoft has not shown by a preponderance of the evidence that claim 34 would have been obvious over Chang and Dickerson.

3. Claim 35

For claim 35, which depends from claim 34, Microsoft relies on the combination of Chang, Dickerson, and Crus. Pet. 49–50. Microsoft

contends that Crus teaches "reciprocal pointers to said two logical columns" as required by claim 35. *Id.* Chang and Dickerson cannot be combined to meet claim 35 for the same reasons as set forth above with respect to claim 34. Accordingly, we conclude that Microsoft has not shown by a preponderance of the evidence that claim 35 would have been obvious over Chang, Dickerson, and Crus.

F. Obviousness over Chang and Goldberg

For the reasons given below, after consideration of Microsoft's Petition and Enfish's PO Response, and the evidence cited therein, Microsoft has not shown, by a preponderance of the evidence, that claim 36 is unpatentable as obvious over the combination of Chang and Goldberg.

1. Goldberg

Goldberg teaches a database management system for a relational database which facilitates storage and retrieval of what are known as "directed graphs." Ex. 1011, 1:7–10, Fig. 1. Goldberg also discloses that a column in a table will have a reference data type, and that entries in that column "are pointers to rows in a specified table." *Id.* at Abst. Goldberg further explains that the directed graph structure is stored as a record/row in a table, "with references corresponding to interconnections between records being stored in reference data type columns." *Id.*

2. Claim 36

Microsoft relies on Dr. Hosking's Declaration to explain how each limitation of claim 36 is disclosed in Chang and Goldberg. Pet. 50–51 (citing Ex. 1022 ¶¶ 227, 310–313). In the Petition, Microsoft addresses only the second limitation in claim 36, not the first limitation. *Id.* In Dr. Hosking's Declaration, however, he states that Chang discloses, "at least one

of said plurality of records includes information defining the type of a different record," addressing the first limitation recited in claim 36. With respect to this first limitation, Dr. Hosking testifies that:

> 310. As previously stated, Chang discloses a Packed Description, PD, in a row that contains information about columns that are represented by rows. *See* Claim 1[E] and 31[E] Anticipation by Chang Discussion. The PD includes information that defines the column type. *Id.* The column is represented as a row in the SYS.COLUMNS table. *Id.* Accordingly, the PD disclosed by Chang includes information that defines the type of a different logical row.

Ex. 1022 ¶ 310.

Enfish argues that Dr. Hosking's rationale is flawed in that it relies upon the Packed Description in SYS.TABLES table, and the column defining the Packed Description in SYS.COLUMNS table, and therefore does not meet the single table requirement of base claim 31. PO Resp. 46–47. Enfish contends that the correct reading claims 31 and 36, "[t]he 'at least one of said plurality of logical rows' and the 'different logical row' must be within <u>the same table</u>." *Id.* at 46.

Microsoft's assertion is based upon the Packed Description in SYS.TABLES containing the definitions of column types of any particular table, and, that the columns of SYS.TABLES are represented as a row in the SYS.COLUMNS table, thus "the PD disclosed by Chang includes information that defines the type of a different logical row." Pet. 50 (citing Ex. 1022 ¶ 310). We are persuaded that this evidence including both the SYS.COLUMNS and SYS.TABLES tables, even when read as Microsoft suggests, does not meet the single table requirement of claims 31 and 36 as discussed in section III.A.3. Accordingly, we conclude that Microsoft has

not shown by a preponderance of the evidence that claim 36 would have been obvious over Chang and Goldberg.

G. Obviousness over Chang and Anderson

For the reasons given below, after consideration of Microsoft's Petition, Enfish's Preliminary Response, and PO Response, and the evidence cited therein, Microsoft has not shown, by a preponderance of the evidence, that claim 37 is unpatentable as obvious over the combination of Chang and Anderson.

1. Anderson

Anderson discloses an electronic spreadsheet having numbered columns, for example 1, 2, 3 . . . etc., and letters referencing columns, for example A, B, C . . . etc. Ex. 1012, 1:51–65. The intersection of any row and column, for example B2, defines an addressable storage location, i.e. a "cell" for holding text and numeric information. *Id.* Anderson also teaches that spreadsheet cells can store formulas applying calculations to numbers stored in spreadsheet cells. *Id.* at 2:3–6. Anderson explains that "[i]n this fashion, cell references can serve as variables in an equation, thereby allowing precise mathematical relationships to be defined between cells." *Id.* at 6–9.

2. Claim 37

In support of this asserted ground of unpatentability, Microsoft relies on the Hosking Declaration (Ex. 1022 ¶¶ 228, 317–325), explaining how each limitation is disclosed in Chang and Anderson. Pet. 51–52.

Microsoft asserts that Anderson teaches a spreadsheet program having notebooks, i.e., tables, with cells, rows and columns, where the cells have pointers to other columns. Pet. 51 (citing Ex. 1012, Figs. 2–5). Dr. Hosking

testifies that one of skill in the art would understand spreadsheet programs as a simple form of a database similar to relational databases. Ex. 1022, ¶ 321, Further, Dr. Hosking states that one of skill in the art would therefore recognize it as obvious to combine the cell pointer functions in such spreadsheets with the relational tables in Chang, so that a cell in Chang's table would contain a plurality of pointers to other columns which contain defined values. *Id.* ¶ 321–325 (citing Ex. 1012, Figs. 4J, 4G).

Enfish argues that a cell address in a spreadsheet, for example the intersection of column A and row 2, "A2," shown in Anderson's Figure 4H, is not a "pointer" in a relational database, but "a cell reference [that] merely serves as a variable in the formula and the data stored within this referenced cell (i.e., cell A2) is used in the calculation." Prelim. Resp. 46. Enfish asserts specifically that "[t]he term 'pointer' is known in the art to mean a variable that stores an address to a location where an object resides. The formula variables in Anderson are not pointers." PO Resp. 48. Enfish relies upon Dr. Jagadish to explain that one of ordinary skill in the art of relational databases would not have understood a cell reference in Anderson's spreadsheet program, to be a "pointer," in a relational database as recited in the '604 patent. Ex. 2007 ¶ 212. Dr. Jagadish refers to three textbooks which explain that [t]he plain and ordinary meaning of the term "pointer" is a variable that stores the address where another object resides." *Id.* ¶ 207. We credit Dr. Jagadish's testimony and evidence as to the meaning of the word "pointer" as it is understood by one of skill in the art because it is consistent with usage of the term "pointer" in the '604 patent. The '604 patent, although it does not give a specific definition, describes for example an OID used as a pointer to an object, where "the 'Employed By' column

126 is synchronized with the 'Employees' column by an OID pointer in the 'Synchronize With' column 144 to the 'Employees' column, represented by row 139." Ex. 1001, 10:8–12, *see also, Id.* at 12: 22–32.

Microsoft does not explain sufficiently or provide evidence that a cell reference for incorporating a variable stored in the cell into a formula in a spreadsheet would have been understood as a variable that stores an address to a location where an object resides in the database, at the time of the invention of the subject matter of the '604 patent.

We conclude that Microsoft has not shown by a preponderance of the evidence that claim 37 would have been obvious over Chang and Anderson.

H. Obviousness over Chang and Covey

For the reasons given below, after consideration of Microsoft's Petition and Enfish's Preliminary Response, and the evidence cited therein, Microsoft has not shown, by a preponderance of the evidence, that claim 40 is unpatentable as obvious over the combination of Chang and Covey.

1. Covey

Covey is titled "Method for Creating and Maintaining a Database for a Dynamic Enterprise" and discloses a method of constructing a database which records a "token" to identify a database object, where a token is created for each event and includes a field corresponding to the event date. Ex. 1014. Abst. The database thus creates a record that preserves the historical state conditions of the database. *Id.* at 8:21–26.

2. Claim 40

Microsoft argues that dependent claim 40 is obvious in view of Chang and Covey. Microsoft concedes that Chang fails to disclose an OID having a session identification number and a timestamp. Pet. 53. Microsoft asserts

that claim 40 is made obvious by Covey's disclosure of a relational database associating time values and session identifiers with database objects. *Id.* at 53–54 (citing Ex. 1014, Abst., Figs. 12–13, 202). Microsoft argues that "[b]ecause Chang and Covey both address the same technical issues and disclose closely related subject matter, a person having ordinary skill in the art would be motivated to combine Chang and Covey." *Id.* at 54.

Enfish counters that Microsoft's statement that the references are analogous art is not sufficient to support obviousness, and that Microsoft's assertions are unsupported by any reliance on Dr. Hosking's Declaration. Prelim. Resp. 50. Microsoft does not, in the Petition, provide any reason for the combination of Chang and Covey apart from asserting the references are analogous art and address similar technical issues. Pet. 50. This is not a sufficient reason with rational evidentiary underpinnings to support the combination of Chang and Covey.

We conclude that Microsoft has not shown by a preponderance of the evidence that claim 40 would have been obvious over Chang and Covey.

I. Secondary Considerations

The factual inquiries for obviousness include secondary considerations based on evaluation and crediting of objective evidence. *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966). However, to accord substantial weight to objective evidence requires the finding of a nexus between the evidence and the merits of the claimed invention. *In re GPAC Inc.*, 57 F.3d 1573, 1580 (Fed. Cir. 1995); *see also In re Huang*, 100 F.3d 135, 140 (Fed. Cir. 1996) ("success is relevant in the obviousness context only if there is proof that the sales were a direct result of the unique characteristics of the claimed invention.").

Enfish contends that the claimed invention received industry accolades, including praise from Microsoft, satisfied a long-felt need, resulted in success where others had failed, as well as commercial success and copying. PO Resp. 54–59. Enfish points to features of claim 31 that it contends resulted in the objective indicia of success, such as industry accolades and commercial success, to which Enfish refers. *Id.* at 54. We are not convinced by this argument. Claim 31 of the '604 patent is challenged as anticipated by Chang. Secondary considerations do not weigh into determinations regarding anticipation. *Cohesive Techs., Inc. v. Waters Corp.*, 543 F.3d 1351, 1364 (Fed. Cir. 2008). In addition, we are not persuaded by secondary considerations here because Enfish has not shown a nexus between any of the accolades or successes it says occurred and the use of the pointer or pointer searching function.

Enfish additionally asserts that Microsoft failed at developing a suitable search engine. PO Resp. 58–59. We are not persuaded by this argument. The statements of Bill Gates and others at Microsoft relied on by Enfish in support of this assertion are not tied sufficiently to any claim at issue in this proceeding. Instead, the statements broadly refer to search engines. PO Resp. 58 (citing Ex. 2050, 2055, 2029, 2030). None of the statements reference two-way pointers, folder objects or searching for pointers or OID's.

Enfish further submits evidence to show commercial success. PO Resp. 59. The evidence Enfish cites refers to how many users downloaded Enfish's software and to the planning of a collaborative effort as set forth in one paragraph of its business plan. *See* Ex. 2025, 2. Enfish's evidence, however, does not indicate that these users paid for or actually used the

downloaded software. The evidence also does not indicate how the number of downloads indicates commercial acceptance, for example, as compared to downloads of other software at the time. Additionally, a planned collaborative effort does not indicate the results of the collaboration. Enfish's evidence does not establish commercial acceptance or financial success. *See In re Fielder*, 471 F.2d 640, 644 (CCPA 1973). Thus, we are not persuaded by Enfish's evidence of commercial success.

J. Motion to Correct Patent Owner Response

After institution of trial, Enfish timely filed a Patent Owner Response (Paper 26), along with the Jagadish Declaration (Ex. 2007), which Enfish later redacted (Paper 32). On September 16, 2014, Enfish filed an unopposed motion to file a corrected Jagadish Declaration, which Enfish contends correct only typographical errors and erroneous citations. Paper 40. We grant Enfish's September 16, 2014 motion.

K. Joint Stipulation

On November 14, 2014, the parties filed a joint stipulation requesting that we expunge confidential versions of exhibits 2049–2058 and 2060–2065. Paper 56. The parties contend that Microsoft withdraws its motion to seal (Paper 30) provided that we expunge the confidential versions. Paper 56. Microsoft agrees that the sealed version of Exhibit 2059 may be unsealed. *Id.* We hereby grant the motion and expunge only confidential versions of exhibits 2049–2058 and 2060–2065.

L. Motion to Exclude

On November 3, 2014, Microsoft filed a motion to exclude Exhibit 2071, the Declaration of Dr. Sharad Mehrotra ("Mehrotra Declaration") and two paragraphs of the Declaration of Louise Wannier ("Wannier

Declaration," Exhibit 2077 ¶¶ 32, 33). Paper 49.

Regarding the Mehrotra Declaration, we agree with Microsoft's assertion that Dr. Mehrotra provides only conclusory opinions and, therefore, we do not rely on it in this Decision. 37 C.F.R. § 42.65(a). Because Microsoft has not argued persuasively any other reason to exclude the Mehrotra Declaration, we deny Microsoft's request to exclude it.

Regarding the Wannier Declaration, we disagree with Microsoft that "Patent Owner has no basis to file the Wannier Declaration as supplemental evidence because Microsoft has not moved to exclude the Armon Declaration." Paper 49, 4–5. Patent Owner is entitled to submit supplemental evidence in response to Microsoft's objection. 37 C.F.R. § 42.64(b)(2). Microsoft further contends that the Wannier Declaration inserts untimely, conclusory, and improper technical opinions. Paper 49, 5. Patent Owner contends that paragraphs 32 and 33 do not exceed the scope because they are submitted to support admissibility. Paper 59, 3–4. We agree with Microsoft that the Wannier Declaration provides conclusory technical opinions, and, therefore, do not rely upon it. Because Microsoft has not argued persuasively any other reason to exclude paragraphs 32 and 33 of the Wannier Declaration, we deny Microsoft's request to exclude it.

IV. CONCLUSION

We conclude that Microsoft has demonstrated by a preponderance of the evidence that (1) claim 31 of the '604 patent is anticipated by Chang, and (2) claims 41 and 42 of the '604 patent are obvious over the combination of Chang and Horn, or Chang and Webb.

We further conclude that Microsoft has not shown that claims 32–40, and 43– 45 of the '604 patent are unpatentable as obvious. In addition, we

terminate this proceeding with respect to claims 1–15 under 37 C.F.R. § 42.72.

IV. ORDER

For the reasons given, it is

ORDERED that claims 31, 41, and 42 of U.S. Patent No. 6,151,604 are determined by a preponderance of the evidence to be unpatentable;

FURTHER ORDERED that this proceeding is TERMINATED, under 37 C.F.R. § 42.72, with respect to claims 1–15;

FURTHER ORDERED Enfish's motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration (Paper 38) is GRANTED;

FURTHER ORDERED that Microsoft's motion to exclude (Paper 48) is DISMISSED;

FURTHER ORDERED that confidential versions of Exhibits 2049–2058 and 2060–2065 are EXPUNGED;

FURTHER ORDERED Microsoft's motion to seal is DISMISSED; and

FURTHER ORDERED that because this is a final written decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

For PETITIONER:

Amy E. Simpson
Chad Campbell
PERKINS COIE LLP
ASimpson@perkinscoie.com
CCampbell@perkinscoie.com

For PATENT OWNER:

Frank Pietrantonio
Orion Armon
COOLEY LLP
fpietrantonio@cooley.com
oarmon@cooley.com
zpatdcdocketing@cooley.com

PTAB Decision in IPR2013-00563

Trials@uspto.gov
571.272.7822

Paper 65
Entered: March 2, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

ENFISH, LLC,
Patent Owner.

Case IPR2013-00563
Patent 6,151,604

Before THOMAS L. GIANNETTI, BRYAN F. MOORE, and BARBARA A. PARVIS, *Administrative Patent Judges.*

PARVIS, *Administrative Patent Judge.*

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.73

I. INTRODUCTION

A. Background

Microsoft Corporation ("Microsoft") filed a petition to institute an *inter partes* review of claims 16–30 and 46–60 of U.S. Patent No. 6,151,604 ("the '604 patent"). Paper 1 ("Pet."). We instituted trial for claims 16–26, 30, 46–56, and 60 of the '604 patent on certain grounds of unpatentability alleged in the Petition. Paper 14 ("Decision to Institute" or "Inst. Dec.").

After institution of trial, Patent Owner, Enfish, LLC ("Enfish"), filed a Patent Owner Response, along with a Declaration by Dr. H.V. Jagadish ("Jagadish Declaration").

On September 16, 2014, Enfish filed an unopposed motion to correct typographical errors in both papers and filed therewith the corrected Patent Owner Response ("PO Resp.," Paper 42) and the corrected Jagadish Declaration (Ex. 2007). We grant Enfish's motion to correct both papers.[1] Microsoft filed a Reply ("Pet. Reply"). Paper 37.

A consolidated hearing for IPR2013-00559, IPR2013-00560, IPR2013-00561, IPR2013-00562, and IPR2013-00563 was held on December 3, 2014. The transcript of the consolidated hearing has been entered into the record. Paper 64 ("Tr.").

We have jurisdiction under 35 U.S.C. § 6(c). This final written decision is issued pursuant to 35 U.S.C. § 318(a).

Microsoft has shown by a preponderance of the evidence that claims 46–50 and 54 of the '604 patent are unpatentable. Microsoft has not shown

---

[1] For ease of reference, throughout we refer to Enfish's corrected Patent Owner Response (Paper 42) and corrected Jagadish Declaration (Exhibit 2007), both filed on September 16, 2014.

that claims 51–53, 55, 56, and 60 of the '604 patent are unpatentable. For the reasons discussed below, for claims 16–26 and 30, we are unable to reach a determination on the alleged grounds of unpatentability over prior art. Accordingly, we terminate this proceeding with respect to claims 16–26 and 30 under 37 C.F.R. § 42.72.

B. Additional Proceedings

In addition to this petition, Microsoft has filed a petition challenging the patentability of claims 1–15 and 31–45 of the '604 patent. *See Microsoft Corp. v. Enfish, LLC*, IPR2013-00562. Microsoft indicates that claims 1, 2, 16, 17, 31, 32, 46, and 47 of the '604 patent have been asserted against Microsoft in *Enfish LLC v. Microsoft Corporation, et al.*, Case No. 12-cv-7360 MRP, in the Central District of California ("California case"). Pet. 2. Microsoft further contends that a final judgment against Enfish has been entered in the California case. Ex. 1168.

Microsoft also has filed three petitions challenging the patentability of claims 1–60 of U.S. Patent 6,163,775 ("the '775 patent").[2] *See Microsoft Corp. v. Enfish, LLC*, IPR2013-00559; IPR2013-00560; IPR2013-00561.

C. The '604 Patent

The '604 patent (Ex. 1101) is titled, "Method and Apparatus for Improved Information Storage and Retrieval System," and generally relates to a system and method for data storage, manipulation, and retrieval in a logical table of a database. This table is a logical structure, not a physical structure, stored in the memory. Ex. 1101, 6:31–33.

[2] The '604 patent and the '775 patent both issued from continuations of application No. 08/383,752, filed March 28, 1995, now U.S. Patent No. 5,729,730. Pet. 6.

Figure 3 is reproduced below:

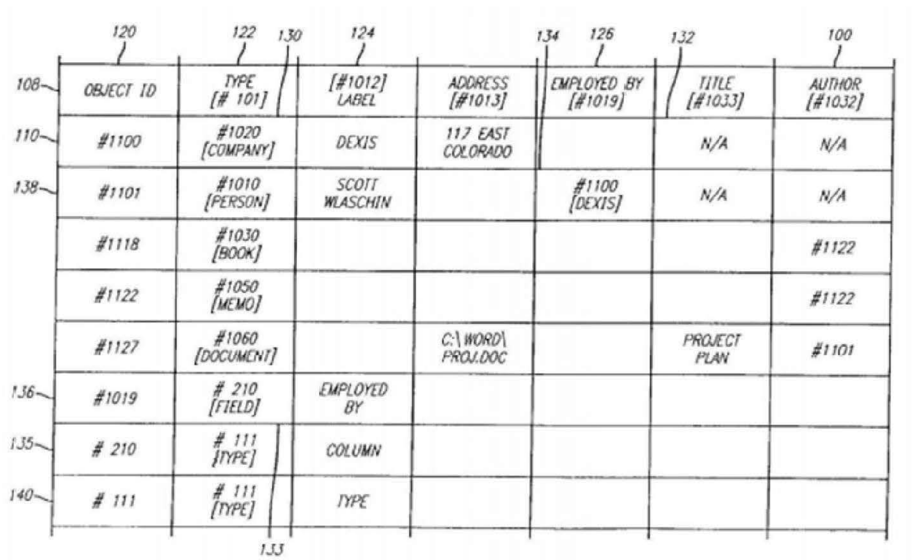


Figure 3 of the ’604 patent illustrates a structure of a logical table. Ex. 1101, 3:30–31. As depicted by Figure 3 of the ’604 patent, above, table 100 is defined by rows 108, 110, 138, 136, 135, and 140, and columns 120, 122, 124, and 126. *Id.* at Fig. 3. The intersection of a row and a column defines a cell in the table. *Id.* at 6:40–41. Each column corresponds to an attribute spanning various records. *Id.* at 6:38. An attribute is a single class description, such as an employer, denoted in column 126 of Figure 3, for example, by the text “Employed By.” *Id.* at 7:16–18.

Each row corresponds to a record spanning various attributes. Ex. 1101, 6:37–38. For example, row 110 corresponds to a company as shown in cell 130. *Id.* at 6:54–55.

D. Illustrative claims

Of the challenged claims, the independent claims are 16, 17, 24, 46, 47, and 54. Each of claims 18–23 depend, directly or indirectly, from claim 17. Claims 25, 26, and 30 depend, directly or indirectly, from claim 24.

Claims 48–53 depend, directly or indirectly, from claim 47. Claims 55, 56, and 60 depend, directly or indirectly, from claim 54. Claims 16, 46, and 47 illustrate the claimed subject matter and are reproduced below:

> 16. A data storage and retrieval system for a computer memory, comprising:
>
> means for configuring said memory according to a logical table, said logical table including:
>
> a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information; and
>
> a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column, wherein said OID's are variable length.

> 46. A method for storing and retrieving data in a computer memory, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information; and
>
> a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column, wherein said OID's are variable length.

> 47. A method for storing and retrieving data in a computer memory, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:

> a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;
>
> a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and
>
> indexing data stored in said table.

E. The Prior Art References Supporting Alleged Unpatentability

Microsoft relies on the following references:

MICROSOFT® VISUAL BASIC™ PROGRAMMING SYSTEM FOR WINDOWS™ VERSION 3.0 (1993) ("Visual Basic," Ex. 1104).

American National Standard for Information Systems – Database Language – SQL (ANSI INCITS 135-1992 (R1998) ("SQL-92," Ex. 1105).

Jensen et al., U.S. Patent No. 5,615,362 (issued Mar. 25, 1997) ("Jensen," Ex. 1106).

GERARD SALTON & MICHAEL J. MCGILL, INTRODUCTION TO MODERN INFORMATION RETRIEVAL (1983) ("Salton," Ex. 1107).

Smith et al., U.S. Patent No. 5,181,162 (issued Jan. 19, 1993) ("Smith '162," Ex. 1108).

Sudarshan Chawathe et al., *The TSIMMIS Project: Integration of Heterogeneous Information Sources* (Dep't Computer Sci. Stanford Univ.) ("Chawathe," Ex. 1116).

F. The Pending Grounds of Unpatentability

The following chart summarizes Microsoft's patentablilty challenges:

Reference[s]	Basis	Claims Challenged
Visual Basic	§ 102(b)	17, 18, 24, 47, 48, and 54
SQL-92 and Chawathe	§ 103	16 and 46
Visual Basic and Jensen	§ 103	19, 20, 49, and 50
Visual Basic and Salton	§ 103	23, 25, 26, 30, 53, 55, 56, and 60
Visual Basic, Jensen, and Salton	§ 103	21 and 51
Visual Basic and Smith '162	§ 103	22 and 52

In support of the above-referenced grounds of unpatentability, Microsoft relies on the Declaration of Dr. Hosking (Ex. 1119, "Hosking Declaration").

II. CLAIM CONSTRUCTION

A. Legal Standard

In an *inter partes* review, claim terms in an unexpired patent are interpreted according to their broadest reasonable construction in light of the specification of the patent in which they appear. 37 C.F.R. § 42.100(b); *see also In re Cuozzo Speed Techs., LLC.*, No. 14-01301, slip op. at 16, 19 (Fed. Cir. Feb. 4, 2015) ("Congress implicitly adopted the broadest reasonable interpretation standard in enacting the AIA," and "the standard was properly adopted by PTO regulation."). Under the broadest reasonable construction

standard, claims are to be given their broadest reasonable interpretation consistent with the specification, and the claim language should be read in light of the specification as it would be interpreted by one of ordinary skill in the art. *In re Am. Acad. of Sci. Tech. Ctr.*, 367 F.3d 1359, 1364 (Fed. Cir. 2004). Also, we must be careful not to read a particular embodiment appearing in the written description into the claim, if the claim language is broader than the embodiment. *See In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993) ("[L]imitations are not to be read into the claims from the specification.").

B. Overview of the Parties' Positions

Microsoft contends that this case involves a computer-implemented invention. Pet. 6. Microsoft also contends that the challenged independent claims invoke means-plus-function claiming, but the '604 patent fails to disclose an algorithm, which is an issue under § 112, sixth paragraph, that cannot be addressed in this proceeding. Pet. 10. In the Decision to Institute, we invited Enfish to direct us to the specific portions of the specification that clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. Inst. Dec. 11. For the reasons discussed below, we determine that Enfish in its response does not identify sufficient corresponding structure, as required under 35 U.S.C. § 112, sixth paragraph, for the "means for configuring said memory according to a logical table," recited in claims 16, 17, and 24.

Regarding terms in the method claims, in the Decision to Institute, we provided constructions for "logical table" and "object identification number," which are shown in the table below. Inst. Dec. 11–13. We also determined that no express construction is needed for the following terms:

"memory," "logical row," and "logical column." Inst. Dec. 14.

Claim Term or Phrase	Construction in the Decision to Institute
"logical table"	"[W]e construe the term "table" to mean: 'a structure of a database comprising rows and columns.'" Inst. Dec. 12. "We determine no express construction of 'logical' is needed for this decision." *Id.*
"object identification number"	"[W]e construe 'identification number' in light of the specification to mean: 'an array of bits that define.' For the purpose of this decision, an express construction of 'object' is not necessary." *Id.* at 13.

Enfish contends that our construction for "object identification number" is incomplete. PO Resp. 16. We evaluate Enfish's contention below. Enfish also provides contentions regarding two other terms: "anchor," recited in claims 21 and 51, and "external documents," recited in claims 30 and 60. PO Resp. 23. Because we determine that Microsoft has not met its burden with respect to these claims for other reasons, no construction of these terms for the purposes of this Decision is necessary.

We discern no reason, based on the complete record now before us, to change our constructions in the Decision to Institute of "logical table" and "object identification number."

C. Analysis of the Parties' Claim Construction Positions

1. Means for configuring said memory according to a logical table

Independent claims 16, 17, and 24 include the limitation "means for configuring said memory according to a logical table." In the Decision to Institute, we agreed with Microsoft that under the broadest reasonable interpretation, the function for the means for configuring is "configuring

memory according to a logical table." Inst. Dec. 11. Additionally, we considered the corresponding structure for the recited function as including a general purpose computer. *Id.* Enfish does not challenge persuasively either of these determinations; however, Enfish identifies portions of the specification that Enfish contends provide algorithmic support for the recited function. PO Resp. 24–26. In particular, Enfish contends that the '604 patent discloses a four-step algorithm that is linked to the recited function of configuring memory according to a logical table. *Id.*

"[T]he corresponding structure for a § 112 ¶ 6 claim for a computer-implemented function is the algorithm disclosed in the specification." *Aristocrat Techs. Austl. Party Ltd. vs. Int'l Game Tech.*, 521 F.3d 1328, 1333 (Fed. Cir. 2008) (quoting *Harris Corp. v. Ericsson Inc.*, 417 F.3d 1241, 1249 (Fed. Cir. 2005)). Additionally, specific portions of the specification must clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. *See Medical Inst. & Diag. Corp. v. Elektra AB*, 344 F.3d 1205, 1211 (Fed. Cir. 2003). For the reasons set forth below, we conclude that the four steps and other '604 patent specification portions identified by Enfish do not describe an algorithm for the recited function of "configuring memory according to a logical table." We further conclude that Enfish has not clearly linked or associated the recited function to the four steps or the portions of the '604 patent specification that Enfish identifies.

For algorithmic support, Enfish identifies disparate excerpts of the '604 patent specification, which do not link or associate clearly a computer program or algorithm to the function corresponding to the claimed means for configuring said memory according to a logical table. For example, the first

step, "[c]reate, in a computer memory, a logical table" (PO Resp. 24) appears to be similar to the recited function of configuring memory according to a logical table, but the first step is not found in the '604 patent specification. Additionally, none of the portions of the '604 patent specification that Enfish cites for this first step provide an algorithm or computer program for performing the recited function of "configuring memory according to a logical table" or clearly link or associate any algorithm or program to this recited function. PO Resp. 24 (citing Ex. 1101, Abstract, 2:53–57, 6:30–38, Figs. 1, 3, 9). These portions describe that an already-formed table has rows and columns, without describing how memory is configured to create a logical table having rows and columns. Ex. 1101, 2:53–57. One portion states that memories "need not store" the table contiguously, but fails to describe an algorithm or computer program for configuring memory such that a logical table is not stored contiguously. *Id.* at 6:30–38.

Enfish's three remaining steps and the other '604 patent specification portions identified by Enfish fail to remedy these deficiencies. The second step is: "[a]ssign each row and column an object identification number (OID) that, when stored as data, can act as a pointer to the associated row or column." PO Resp. 25. One of the portions of the '604 patent specification cited by Enfish for the second step indicates "the system must generate a unique OID when columns and rows are formed." Ex. 1101, 8:6–7. The remainder of the identified portions relate to assigning an OID or indicate that an OID "may be used" as a pointer, without describing an algorithm for forming columns and rows of a table or showing how assigning an OID relates to steps for forming columns and rows. Ex. 1101, Abstract, 2:54–55,

6:42–49, 6:60–61, 8:10–51, Figs. 3, 4. Additionally, the identified portions discuss assigning a numeric value to an OID in the form of a bit array, but fail to describe how to configure memory such that an OID may be used as a pointer. *Id.*

The third step is a feature of claim 1 (PO Resp. 12). That claim is not challenged in this proceeding. Furthermore, the portions of the '604 patent specification cited by Patent Owner for the third step (PO Resp. 26 (citing Ex. 1101, Abstract, 2:60–63, 6:39–40, 7:16–22, Fig. 3)) indicate that a table has a row that corresponds to columns, such as a header row, which is a generic feature of an already-formed table. These '604 patent specification portions identified by Enfish (*id.*), however, do not describe how to form a table with this feature or link or associate this feature to the recited function.

The fourth step, i.e., storing and accessing data in cells (PO Resp. 26), is performed in an already-configured table and, therefore, occurs after the recited function of configuring a table has occurred. Nonetheless, the '604 patent specification portions identified by Enfish (*id.* (citing Ex. 1101, Abstract, 2:57–60, 6:40–41, 6:53–54, 7:1–2, 7:14–15, 11:47–59, Fig. 10)) suffer from the same deficiencies noted above.

We conclude that the four steps and '604 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table. The portions of the '604 patent specification identified by Enfish describe an already-formed table having generic characteristics, such as columns, rows, identifiers, and a header row. *See, e.g.*, Ex. 1101, Fig. 3. The description, however, does not disclose any algorithm or computer program for forming this table. Additionally, the description identified by Enfish is not linked or associated clearly with the

recited function of "configuring memory according to a logical table."

Enfish also relies on Dr. Jagadish's Declaration to show that the '604 patent specification provides an algorithm and clearly links the algorithm to the recited function. PO Resp. 24–26 (citing Ex. 2007 ¶¶ 68–74). The Jagadish Declaration, however, relies on similar portions of the '604 patent specification to those cited in Patent Owner's Response. For the reasons given, the Jagadish Declaration does not support Enfish's assertion. Dr. Jagadish further asserts, "[i]t is also my opinion that one of ordinary skill in the art would understand how to implement those algorithm steps using techniques and resources that were available at the time the '775 Patent was filed." Ex. 2007 ¶ 69. Enfish, however, cannot rely on the knowledge of one skilled in the art to address the deficiencies noted above. *See Function Media, LLC v. Google Inc.*, 708 F.3d 1310, 1319 (Fed. Cir. 2013) ("Having failed to provide any disclosure of the structure . . . FM cannot rely on the knowledge of one skilled in the art to fill in the gaps.")

Enfish, furthermore, identifies the same algorithmic support for each of independent claims 16, 17, and 24 (PO Resp. 22–24). We are not persuaded by this argument because each of these claims recites a different table. Enfish contends that these differences are addressed because certain features in the algorithm are identified as optional. Tr. 43:1–43:23. This argument is not convincing because Enfish does not specify the claim or claims supported by each of these optional features. Furthermore, the '604 patent specification portions identified by Enfish do not disclose a computer program or algorithm for configuring memory according to a logical table, regardless of whether so-called optional features are included.

Because we conclude that the four steps and '604 patent specification

portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table, we terminate this proceeding with respect to the claims that recite this means-plus-function limitation. As explained in *BlackBerry Corporation v. Mobile Media Ideas, LLC*, Case IPR2013-00036 (PTAB Mar. 7, 2014) (Paper 65), the specification must disclose enough of a specific algorithm to provide the necessary structure under § 112, sixth paragraph. In the circumstance when the specification of the challenged patent lacks sufficient disclosure of structure under 35 U.S.C. § 112, sixth paragraph, the scope of the claims cannot be determined without speculation and, consequently, the differences between the claimed invention and the prior art cannot be ascertained. *Id.* For the reasons given, we determine that independent claims 16, 17, and 24 are not amenable to construction and, thus, we terminate this proceeding with respect to these claims under 37 C.F.R. § 42.72. Because claims 18–23 depend, directly or indirectly, from claim 17, and claims 25, 26, and 30 depend, directly or indirectly, from claim 24 we also terminate this proceeding with respect to these claims under 37 C.F.R. § 42.72.

2. Other Means-Plus-Function Terms

Because we determine that this proceeding should be terminated as to claims 16–26 and 30 for the reasons discussed above, we need not construe other means-plus-function terms appearing in those claims for the purposes of this Decision.

3. Object Identification Number (OID)

Independent claims 46, 47, and 54 recite "object identification number (OID)" and "OID." In the Decision to Institute, we construed "identification

number" in light of the specification to mean: "an array of bits that define." Inst. Dec. 13 (citing Ex. 1101, 3:32–34, 8:35–37). We also decided that an express construction of "object" is not necessary.

Enfish "agrees in principle" with our construction, but contends that the construction is incomplete. PO Resp. 16. Enfish contends that "OID" should be defined further as "a unique, immutable, and system-generated value that identifies an object." *Id.* at 15. Microsoft contends that Enfish seeks to read into the claims extraneous limitations that are unsupported by either intrinsic or extrinsic evidence. Pet. Reply 2. Microsoft additionally contends that Enfish's proposed construction does not provide an appropriate context for the additional limitations. *See, e.g.*, Pet. Reply 3.

At the heart of Enfish's contention is an assertion that an OID will not function properly if it is not unique, immutable, and system generated. PO Resp. 17–22. For example, Enfish, relying on Dr. Jagadish, contends that an OID must be a unique value because "if an OID were not unique the database would be non-functional because objects could not be reliably retrieved." PO Resp. 17 (citing Ex. 2007 ¶ 51). Dr. Jagadish cites to a portion of the specification stating that the OID is used for "exact retrieval" and states that without uniqueness, retrieval using an OID will result in more than one object. Ex. 2007 ¶ 51 (citing Ex. 1101, 1:60–64).

We determine that Enfish's proposed construction of OID introduces requirements beyond what is needed for an OID to identify data and retrieve data from memory. Enfish, for example, contends that "an OID is unique among all rows and columns of the logical table." PO Resp. 32. Additionally, each of the independent claims recites that an OID is included "to identify" each logical row and each logical column, but does not

specifically recite using an OID to retrieve data from memory.

Microsoft advances additional contentions regarding why Enfish's proposed construction should not be adopted. For example, Microsoft contends that Enfish's "unique" OID requirement would conflict with the patent, which describes a row in a table with an OID that has the same value as the OID of a column in the same table. Pet. Reply 3 (citing Ex. 1101, Fig. 3). Microsoft also contends that "immutable" should not be imported into the construction of OID because the term is absent from the specification, claims, and file history, and the specification is devoid of disclosure relating to immutability of an OID. Pet. Reply 3–4. As further support, Dr. Hosking states that an OID may change in certain circumstances and still retrieve information reliably. Ex. 1143 ¶¶ 21, 22.

We determine that adopting Enfish's proposed construction without further context would create ambiguity. Additionally, limitations are not to be read into the claims from embodiments in the specification. *See In re Van Geuns*, 988 F.2d at 1184. Furthermore, we agree with Microsoft that Enfish's reliance on extrinsic evidence is misplaced. Enfish, for example, relies on excerpts of "text books" (PO Resp. 21) for its contention that an OID is "immutable," without showing sufficiently that the definition ascertained from these excerpts of extrinsic evidence is consistent with the definition that would be ascertained by reading the patent documents.

We, therefore, determine that the construction of "OID" in the Decision to Institute should not be changed.

III. ANALYSIS

A. Alleged Anticipation of Claims 47, 48, and 54 by Visual Basic

For the reasons given below, after consideration of the arguments in Enfish's Patent Owner Response, and the evidence cited therein, we conclude that Microsoft has shown, by a preponderance of the evidence, that each of claims 47, 48, and 54 is unpatentable as anticipated by Visual Basic.

1. Visual Basic

Visual Basic describes a programming system for the Windows™ operating system that enables programmers to create databases. Ex. 1104, PF 1, PG 453.[3] The programming system uses objects to represent tables of a database. *Id.* at PF 47. As described in Visual Basic, a table object is a logical representation of a physical table with records (rows) and fields (columns). *Id.* at LR 558.

Visual Basic also describes adding an index to a database, "[a]dding an index to your database can increase the speed with which you get access to information." Ex. 1104, PF 37. Visual Basic provides an example of an index with key fields Name, Last Name, City, and Zip Code. *Id.* at 38.

2. Claims 47 and 54

We have reviewed Microsoft's anticipation argument, supporting evidence, and the detailed claim analysis, which reads persuasively all elements of each of claims 47 and 54 onto the disclosure of Visual Basic.

---

[3] Citations in this decision to Exhibit 1104 refer to the volumes Programmer's Guide (PG), Language Reference (LR), and Professional Features (PF), respectively, as well as the page number within each volume.

Pet. 31–43 (citing Ex. 1104, LR 162, 185, 276, 277, 280, 532, 425, 558, PF 1, 3, 18, 37–40, 47, 76, 80, PG 471, 481, 482, 484; Ex. 1119 ¶¶ 68–93, 119). For example, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses configuring memory according to a logical table, as recited in claims 47 and 54. In the cited portions above, Visual Basic discloses storing data in a computer formatted as a table object, which is "a logical representation of a physical table." Ex. 1104, LR 558. The table object comprises "records (rows) and fields (columns)." *Id.*

As an additional example, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses each row and each column including an OID, as recited in claims 47 and 54. In the cited portions above, Visual Basic describes assigning a primary key and that "[a] table's primary key is the determining factor when testing to see if the record is unique within the table." Ex. 1104, PF 39. According to Visual Basic "[Visual Basic] creates an index on the primary key of the table and uses it to find records and to create joins between tables." *Id.* Regarding each column including an OID, according to Visual Basic, "[e]ach Field in the Fields collection of a TableDef has a unique value in the OrdinalPosition property." *Id.* at LR 425.

Furthermore, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses indexing data stored in the table, as recited in claims 47 and 54, and a cell including a pointer to an index record, as recited in claim 54. In the cited portions above, Visual Basic discloses an index object used for creating indexes to "speed up the process of finding records." Ex. 1104, LR 276–277; *see also*

id. at PF 37, 38, PG 482. Visual Basic also discloses a cell pointing to an index object. *Id.* at LR 280, PG 484.

Enfish contends that Microsoft has not met its burden of showing that Visual Basic discloses logical column OIDs and logical row OIDs. PO Resp. 30–34. Enfish's contentions are based on its proposed construction of OID, which we decline to adopt for the reasons given above.

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claims 47 and 54 of the '604 patent are anticipated by Visual Basic.

3. Claim 48

Claim 48 depends directly from independent claim 47. Microsoft has shown by a preponderance of the evidence that Visual Basic describes the additional elements recited in claim 48 of searching the table for a key word and inserting into the table a record corresponding to the key word. Pet. 42–43 (citing Ex. 1104, PF 1, 37, PG 487, Fig. 20.6; Ex. 1119 ¶ 91). Accordingly, Microsoft has demonstrated by a preponderance of the evidence that claim 48 of the '604 patent is unpatentable, under 35 U.S.C. § 102(b), as anticipated by Visual Basic.

B. Obviousness of Claim 46 over SQL-92 and Chawathe

For the reasons given below, after consideration of Enfish's Patent Owner Response, and the evidence cited therein, we conclude that Microsoft has shown, by a preponderance of the evidence, that claim 46 is unpatentable as obvious over the combination of SQL-92 and Chawathe.

1. SQL-92

SQL-92 describes a standard specifying the syntax and semantics of

the SQL database language. Ex. 1105, 1. In particular, SQL-92 describes a table object, which is a multi-set of rows. *Id.* at 12, 29; *see also id.* at 82 ("A <table name> identifies a table."). Additionally, the SQL-92 table object comprises columns. *Id.* at 12; *see also id.* at 82 ("A <column name> identifies a column.").

2. Chawathe

Chawathe describes a project for integrating heterogeneous information sources referred to as The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) project. Ex. 1116, 1. Chawathe states, for the TSIMMIS project, an object exchange model (OEM) is adopted. *Id.* Additionally, a SQL-like query language, OEM-QL, is adopted for requesting OEM objects. *Id.* at 2.

3. Claim 46

We have reviewed Microsoft's obviousness argument, supporting evidence, and the detailed claim analysis, which reads persuasively all elements of claim 46 onto the teachings of SQL-92 and Chawathe, taken together. Pet. 43–50 (citing Ex. 1105, 1, 12, 28–30, 82, 156, 193, 220, 312–14, 316, 322, 477, 497; Ex. 1116, 1, 2, 5, 6; Ex. 1119 ¶¶ 125, 127–29, 131, 133–35, 138–40, 143–44, 146–51). For instance, we are persuaded that Microsoft has shown by a preponderance of the evidence that the combination of SQL-92 and Chawathe teaches configuring memory according to a logical table, as recited in claim 46. SQL-92 teaches "the descriptors of enabling objects (e.g., tables) are said to *include* the descriptors of enabled objects (e.g., columns or table constraints)." Ex. 1105, 12. Additionally, SQL-92 teaches "[a] table is a multiset of rows." *Id.*

at 29.

Additionally, we determine that Microsoft has shown by a preponderance of the evidence that the combination of SQL-92 and Chawathe teaches each row and each column including an OID, as recited in claim 46. For instance, SQL-92 teaches assigning a primary key that uniquely identifies a row in a table. Ex. 1105, 32–33, 497. SQL-92 also teaches that each column has an ordinal position. *Id.* at 29. Furthermore, Chawathe teaches a unifying object model, in which each object has a unique variable-length identifier referred to as "Object-ID." Ex. 1116, 5.

Enfish contends that Microsoft's motivation-to-combine arguments are insufficient. PO Resp. 42. We, however, are persuaded that Microsoft has set forth a sufficient articulated reason with a rational underpinning to support obviousness. *See KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 418 (2007) (citing *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006)). For instance, Microsoft relies on Dr. Hosking's Declaration (Ex. 1119 ¶¶ 358–59), which states it would have been obvious to one of ordinary skill in the art to combine the techniques of SQL-92 and Chawathe to use an SQL-like language to query objects because Chawathe suggests combination with SQL-92 to enhance SQL-92 language capabilities. Ex. 1119 ¶ 359 (citing Ex. 1116, 2, 6). Chawathe states "OEM-QL is an SQL-like language extended to deal with labels and object nesting . . . ." Ex. 1116, 2; *see also id.* at 6 ("OEM-QL adapts existing SQL-like languages for object-oriented models . . . to OEM."). We credit Dr. Hosking's statement as it is consistent with the teachings of Chawathe (*see e.g.*, Ex. 1116, 2. 6).

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 46 of the '604 patent is unpatentable as obvious

over the combination of SQL-92 and Chawathe.

C. Obviousness of Claims 49 and 50 in view of Visual Basic and Jensen

For the reasons given below, after consideration of the Petition, Enfish's Patent Owner Response, and the evidence cited therein, Microsoft has shown, by a preponderance of the evidence, that each of claims 49 and 50 is unpatentable as obvious over the combination of Visual Basic and Jensen.

1. Jensen

Jensen describes managing information retrieved from a structured database, such as a relational database, by constructing object instances in which each object instance has its own unique object identifier that provides a mapping between the object instance and at least one row in the structured database. Ex. 1106, Abstract. Jensen further describes a pointer that is an object instance attribute containing an address, such as a physical address, of another object instance. *Id.* at 7:33–35. Jensen also describes bidirectional pointers that are set up between a department instance and related employee instances. *Id.* at 10:9–11.

2. Claims 49 and 50

Claims 49 and 50 depend, directly or indirectly, from claim 48. Microsoft has shown by a preponderance of the evidence that Visual Basic describes the additional elements recited in claims 49 and 50. Pet. 50–52 (citing Ex. 1104, PF 69, PG 487, 489, Fig. 20.6; Ex. 1106, 7:29–35, 8:24–29, 8:50–10:17, 10:45–48, 10:62–11:26, Fig. 2, 3; Ex. 1119 ¶¶ 360–369, 377–377).

Enfish contends that Microsoft's motivation-to-combine arguments

are insufficient. PO Resp. 43. Dr. Jagadish states "[d]ue to the dissimilarity of environments between [Visual Basic] and Jensen (a relational database employing unidirectional pointers in an indexing structure vs. linking related objects in an object cache), extensive redesign of the system in [Visual Basic] would be required." Ex. 2007 ¶ 228. In contrast to Dr. Jagadish's statement, however, Jensen teaches an object-oriented application for managing information retrieved from a structured database, such as a relational database. Ex. 1106, Abstract.

Enfish further contends that Microsoft does not explain why one of ordinary skill in the art would have been motivated to apply the pointers of Jensen to the indexing structure in Visual Basic. PO Resp. 43 (citing Ex. 2007 ¶ 228). We, however, are persuaded that Microsoft has shown a sufficiently articulated reason with rational underpinning to support obviousness. Microsoft states that one of ordinary skill in the art would have been motivated to combine Jensen with Visual Basic considering Jensen's benefits of referencing objects through two-way pointers as an improvement in flexibility (Pet. 51 (citing Ex. 1106, 7:29–35)) as compared to Visual Basic's one-way pointers (Pet. 51 (citing Ex. 1104, PG 487)). Also, Dr. Hosking states that a person of ordinary skill in the art would have been motivated to combine Jensen's teaching of a pointer with Visual Basic's disclosure of index objects. Ex. 1119 ¶¶ 366, 369; *see also id.* ¶ 376 ("It was *well known* in the art at the time that Index Objects locate data by maintaining object identification numbers in the internal pointers as disclosed in Jensen" (emphasis added)). The predictable use of familiar prior art elements according to their established functions renders the recited invention obvious. *See KSR*, 550 U.S. at 417.

For the foregoing reasons, we determine that Microsoft has shown, by a preponderance of the evidence, that claims 49 and 50 of the '604 patent are unpatentable as obvious over the combination of Visual Basic and Jensen.

D. Obviousness of Claims 53, 55, 56, and 60 in view of Visual Basic and Salton and Obviousness of Claim 51 in view of Visual Basic, Jensen, and Salton

For the reasons given below, we conclude that Microsoft has not shown that claims 51, 53, 55, 56, and 60 are unpatentable as obvious.

1. Salton

Salton describes information retrieval. Ex. 1107, 7. In particular, Salton describes speeding up a search for information by developing an index. *Id.* at 16. An index of Salton includes values for each key for records in a file. *Id.* at 17.

2. Claims 53, 55, 56, and 60

Enfish contends that Microsoft's motivation-to-combine contentions are insufficient. PO Resp. 48–51. Dr. Jagadish states, "it is my opinion that a POSITA would not have been motivated to import, into [Visual Basic], the teachings in Salton regarding indexing architectures." Ex. 2007 ¶ 239. Dr. Jagadish provides various reasons including a lack of teachings in Visual Basic, difficulties described in Salton, as well as difficulties that would have been known to one of ordinary skill in the art. Ex. 2007 ¶¶ 239–242.

Microsoft discusses reasons for combining Visual Basic and Salton in three places in the Petition. First, Microsoft states that one of ordinary skill in the art would have been motivated to combine Salton with Visual Basic to employ the indexing and text analysis techniques taught by Salton in the

object database systems of Visual Basic. Pet. 53 (citing Ex. 1119 ¶ 424). We are not persuaded that this is a sufficient rationale for combining Visual Basic and Salton, because Microsoft's citation does not provide adequate support. Microsoft cites only to Dr. Hosking, who states that Visual Basic and Salton are in the same field and provides a conclusion that one of ordinary skill in the art would have been motivated to combine Visual Basic and Salton. Ex. 1119 ¶ 424. Additionally, in its contentions regarding other challenged claims, Microsoft relies on Visual Basic for describing indexing. *See* Pet. 40 (Ex. 1104, PF 37, 38). In light of these other contentions, Microsoft does not explain sufficiently why one of ordinary skill in the art would have been motivated to employ Salton's indexing techniques with Visual Basic.

Second, Microsoft states, "[o]ne of ordinary skill in the art would combine Salton with [Visual Basic] to search for key words and create [an] index for those key words with pointers to the logical cells in [Visual Basic], because both Salton and [Visual Basic] discuss information search and retrieval using computer databases." Pet. 56 (citing Ex. 1119 ¶ 457); *see also* Pet. 56 ("One of ordinary skill in the art would know to combine Salton and [Visual Basic] because both describe storing data in databases, allowing users to query the data, and using indexes to facilitate those queries"). Microsoft's contentions are insufficient because they are based on solely the references teaching the same techniques. Dr. Hosking states, "[o]ne having ordinary skill in the art would know to combine what's described in Salton with what's described in [Visual Basic] because they both deal with solving the problem of locating key words in a database." Ex. 1119 ¶ 442; *see also id.* at ¶ 457 (referencing claim element [25C]). Again, Dr. Hosking's

conclusion is based solely on the references being in the same field. He fails to provide sufficient rational underpinning to explain why a person of ordinary skill would have combined the references.

Third, Microsoft states "[o]ne [of] ordinary skill in the art would know to add the weighting techniques described in Salton to [Visual Basic] to weigh key words and retrieve[] cells." Pet. 58 (citing Ex. 1119 ¶¶ 469–477). In the cited portions of the Hosking Declaration, Dr. Hosking states that "[o]ne having skill in the art would know to combine [Visual Basic] with weigh[t]ing techniques of Salton because both have to do with providing access to data in a database through a search." Ex. 1119 ¶ 473. Although Dr. Hosking indicates that weighting increases the value of search results (*id.* ¶¶ 469–477), he does not indicate that increasing value using weighting is a reason for combining the technology of Salton with Visual Basic to arrive at the claims, which do not recite weighting.

As discussed above, Dr. Jagadish disagrees with Dr. Hosking. Ex. 2007 ¶¶ 239–242. In its Reply, Microsoft does not address Enfish's contentions or the testimony provided by Dr. Jagadish . Pet. Reply 11–12.

3. Claim 51

With respect to claim 51, Microsoft, relying on Dr. Hosking, further states that "[o]ne of ordinary skill in the art would have been motivated to combine Salton with [Visual Basic] and Jensen to employ the indexing and text analysis techniques taught by Salton in the object database systems of [Visual Basic] and Jensen to improve data searching and retrieval efficiency." Pet. 54 (citing Ex. 1119 ¶ 383). Dr. Hosking, however, states that the references relate to data managing and then states his conclusion that one of ordinary skill in the art would have been motivated to combine them.

Ex. 1119 ¶ 383.

4. Conclusion

We determine that Dr. Hosking's statements are not a sufficient rationale for combining these references because he bases his conclusions on simple statements that the references are in the same field. Microsoft does not provide sufficient additional reasons, such as expert testimony that the combinations of prior art references are of familiar elements according to known methods that yield no more than predictable results. We, therefore, conclude that Microsoft has not demonstrated that the teachings would have been combined by a person of ordinary skill.

For the foregoing reasons, we determine that Microsoft has not set forth a sufficient articulated reason with a rational underpinning to support a showing of obviousness by a preponderance of the evidence. We, therefore, conclude that Microsoft has not shown that claims 51, 53, 55, 56, and 60 of the '604 patent are unpatentable as obvious.

E. Obviousness of Claim 52 in view of Visual Basic and Smith '162

For the reasons given below, Microsoft has not shown that claim 52 is obvious.

1. Smith '162

Smith '162 describes an object-oriented document management and production system. Ex. 1108, Abstract. Stored objects are organized, accessed, and manipulated through a database management system. *Id.* For example, documents and folders can be represented as objects. *Id.* at 3:29–35. The hierarchically superior folder object pointers specify documents. *Id.* at 8:19–20.

2. Claim 52

Enfish contends that Microsoft's motivation-to-combine arguments are insufficient. PO Resp. 46. We are not persuaded that Microsoft has set forth a sufficient articulated reason with a rational underpinning to support a determination of obviousness. Microsoft states that one of ordinary skill in the art would have been motivated to combine the folder and document objects of Smith '162 with the objects disclosed in Visual Basic for the benefits of providing varying data architectures in the database. Pet. 54–55 (citing Ex. 1119 ¶ 393). Microsoft, however, relies on Dr. Hosking, who states that one of ordinary skill in the art would have been motivated to combine Visual Basic and Smith '162 because they "address the same technical issues and disclose closely related subject matters." Ex. 1119 ¶ 393. Although Dr. Hosking mentions that Smith '162 relates to managing documents and folders, he does not testify persuasively regarding the benefits of providing varying data architectures in connection with combining Visual Basic and Smith '162. Microsoft does not offer persuasive additional contentions or evidence supporting the asserted combination of Visual Basic and Smith '162.

For the foregoing reasons, Microsoft has not shown that claim 52 of the '604 patent is unpatentable as obvious.

F. Secondary Considerations

We note that factual inquiries for an obviousness determination include secondary considerations based on evaluation and crediting of objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966). Notwithstanding what the teachings of the prior art would have suggested to one with ordinary skill in the art at the time of the

invention, the totality of the evidence submitted, including objective evidence of nonobviousness, may lead to a conclusion that the claimed invention would not have been obvious to one with ordinary skill in the art. *In re Piasecki*, 745 F.2d 1468, 1471–72 (Fed. Cir. 1984). However, to accord substantial weight to objective evidence requires the finding of a nexus between the evidence and the merits of the claimed invention. *In re GPAC Inc.*, 57 F.3d 1573, 1580 (Fed. Cir. 1995); *see also In re Huang*, 100 F.3d 135, 140 (Fed. Cir. 1996) ("success is relevant in the obviousness context only if there is proof that the sales were a direct result of the unique characteristics of the claimed invention.").

Enfish contends that the claimed invention received industry accolades, including praise from Microsoft, satisfied a long-felt need, resulted in success where others had failed, as well as commercial success. PO Resp. 53–57. Enfish points to features of claim 24, which Enfish contends are the features of the claimed invention that resulted in the objective indicia of success to which Enfish refers. *Id.* at 53–54. Claim 24 of the '604 patent, however, is challenged on the basis that it is anticipated by Visual Basic. Secondary considerations do not weigh into determinations regarding anticipation. *Cohesive Techs., Inc. v. Waters Corp.*, 543 F.3d 1351, 1364 (Fed. Cir. 2008).

Independent claim 46, which is challenged on the basis of obviousness, recites some of the same features as claim 24. We, however, determine that SQL-92 teaches all of the features of claim 46, including OIDs, except SQL-92 does not teach explicitly that certain OIDs have variable lengths. Microsoft contends that SQL-92 should be combined with Chawathe because Chawathe teaches each object having a unique variable-

length identifier referred to as "Object-ID." Pet. 50 (citing Ex. 1116, 5). Enfish does not claim to have invented a variable length for an OID. Thus, Enfish has not shown a nexus between any of the accolades or successes it says occurred and the use of the claimed variable length OIDs.

Because we conclude that Microsoft has not shown that claims 51–53, 55, 56, and 60 of the '604 patent are unpatentable for other reasons, the only other claims that we need to evaluate are dependent claims 49 and 50 of the '604 patent. Microsoft asserts that claims 49 and 50 are obvious over the combination of Visual Basic with a reference that teaches referencing objects through two-way pointers. Pet. 51 (citing Ex. 1106, 7:29–35). Enfish has not shown a nexus between any of the accolades or successes it says occurred and the use of two-way pointers.

Enfish additionally submits evidence that Enfish states shows that Microsoft failed at developing a suitable search engine. PO Resp. 58–59. We are not persuaded by this argument. The statements submitted by Enfish are not tied sufficiently to any claim at issue in this proceeding. Instead, the statements broadly refer to search engines. None of the statements reference a variable length for an OID or two-way pointers.

Enfish further submits evidence that Enfish alleges shows commercial success. PO Resp. 59–60. In addition to the shortcomings discussed above, the proffered evidence is further deficient for the following reasons. Enfish contends that the evidence supports that its product was "well received," 75,000 users downloaded Enfish's software, and the functionality of the software resulted in a collaborative effort. *See* PO Resp. 59–60 (citing Ex. 2024; Ex. 2025, 2; Ex. 2030). Enfish's evidence, however, does not indicate that users paid for or actually used the downloaded software. The evidence

also does not indicate how the number of downloads illustrates commercial acceptance, for example, as compared to downloads of other software at the time. Additionally, a planned collaborative effort does not indicate the results of the collaboration. Enfish's evidence does not establish commercial acceptance or financial success. *See In re Fielder*, 471 F.2d 640, 644 (CCPA 1973). Thus, Enfish's evidence is not sufficient to show commercial success.

We, therefore, determine that features of variable length OIDs and pointers would have been predictable variations within the technical grasp of a person of ordinary skill in the art, and are not persuaded by the objective evidence that claims 46, 49, and 50 of the '604 patent are not obvious.

G. Motion to Correct Patent Owner Response

After institution of trial, Enfish timely filed a Patent Owner Response (Paper 26), along with the Jagadish Declaration (Ex. 2007), which Enfish later corrected (Paper 27). On September 16, 2014, Enfish filed an unopposed motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration, which Enfish contends correct only typographical errors and erroneous citations. Paper 40. We grant Enfish's September 16, 2014 request.

H. Joint Stipulation

On November 14, 2014, the parties filed a joint stipulation requesting that we expunge confidential versions of exhibits 2049–2058 and 2060–2065. Paper 57. The parties contend that Microsoft withdraws its motion to seal (Paper 30) provided that we expunge the confidential versions. Paper 55. Microsoft agrees that the sealed version of Exhibit 2059 may be

unsealed. *Id.* We hereby grant the motion and expunge only confidential versions of exhibits 2049–2058 and 2060–2065.

I. Motion to Exclude

On November 3, 2014, Microsoft filed a motion to exclude Exhibit 2071, the Declaration of Dr. Sharad Mehrotra ("Mehrotra Declaration"), and two paragraphs of the Declaration of Louise Wannier ("Wannier Declaration," Exhibit 2077 ¶¶ 32, 33). Paper 50.

Regarding the Mehrotra Declaration, we agree with Microsoft's assertion that Dr. Mehrotra provides only conclusory opinions and, therefore, we give his Declaration no weight and do not rely on it in this Decision. 37 C.F.R. § 42.65(a). Because Microsoft has not argued persuasively any other reason to exclude the Mehrotra Declaration, we dismiss Microsoft's request to exclude it as moot.

Regarding the Wannier Declaration, we disagree with Microsoft that "Patent Owner has no basis to file the Wannier Declaration as supplemental evidence because Microsoft has not moved to exclude the Armon Declaration." Paper 50, 4–5. Patent Owner is entitled to submit supplemental evidence in response to Microsoft's objection. 37 C.F.R. § 42.64(b)(2). Microsoft further contends that the Wannier Declaration inserts untimely, conclusory, and improper technical opinions. Paper 50, 5. Patent Owner contends that paragraphs 32 and 33 do not exceed the scope because they are submitted to support admissibility. Paper 60, 3–4. We agree with Microsoft that the Wannier Declaration provides conclusory technical opinions, and, therefore, give the technical opinions in paragraphs 32 and 33 of her Declaration no weight. Because Microsoft has not argued persuasively any other reason to exclude paragraphs 32 and 33 of

the Wannier Declaration, we dismiss Microsoft's request to exclude it as moot.

IV. CONCLUSION

We conclude that Microsoft has demonstrated by a preponderance of the evidence that (1) claims 47, 48, and 54 of the '604 patent are anticipated by Visual Basic, (2) claim 46 of the '604 patent is obvious over the combination of SQL-92 and Chawathe, and (3) claims 49 and 50 of the '604 patent are obvious over the combination of Visual Basic and Jensen.

We further conclude that Microsoft has not shown that claims 51–53, 55, 56, and 60 of the '604 patent are unpatentable as obvious. In addition, we terminate this proceeding with respect to claims 16–26 and 30 under 37 C.F.R. § 42.72. Claims 27–29 and 57–59 are not at issue in this trial.[4]

This is a final written decision of the Board under 35 U.S.C. § 318(a). Parties to the proceeding seeking judicial review of this decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

---

[4] In the Decision to Institute, we declined to institute an *inter partes* review of claims 27–29 and 57–59 because we were not persuaded that Petitioner had shown that there was a reasonable likelihood of prevailing on its challenges to these claims. Inst. Dec. 2, 25–27.

V. ORDER

For the reasons given, it is

ORDERED that claims 46–50 and 54 of U.S. Patent No. 6,151,604 are determined by a preponderance of the evidence to be unpatentable;

FURTHER ORDERED that this proceeding is TERMINATED, under 37 C.F.R. § 42.72, with respect to claims 16–26 and 30;

FURTHER ORDERED Enfish's motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration (Paper 40) is GRANTED;

FURTHER ORDERED that Microsoft's motion to exclude (Paper 50) is DISMISSED;

FURTHER ORDERED that confidential versions of Exhibits 2049–2058 and 2060–2065 are expunged;

FURTHER ORDERED Microsoft's motion to seal is DISMISSED; and

FURTHER ORDERED that because this is a final written decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

For PETITIONER:

Chun M. Ng
Amy E. Simpson
Chad Campbell
Theodore H. Wimsatt
PERKINS COIE LLP
CNg@perkinscoie.com
ASimpson@perkinscoie.com
CCampbell@perkinscoie.com
TWimsatt@perkinscoie.com

For PATENT OWNER:

Frank Pietrantonio
Orion Armon
COOLEY LLP
fpietrantonio@cooley.com
oarmon@cooley.com
zpatdcdocketing@cooley.com

PTAB Decision in IPR2013-00559

Trials@uspto.gov
571-272-7822

Paper 65
Entered: March 3, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION
Petitioner

v.

ENFISH, LLC
Patent Owner

Case IPR2013-00559
Patent 6,163,775

Before THOMAS L. GIANNETTI, BRYAN F. MOORE, and SCOTT A. DANIELS, *Administrative Patent Judges.*

DANIELS, *Administrative Patent Judge.*

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.73

I. INTRODUCTION

A. Background

Petitioner Microsoft Corporation ("Microsoft") filed a petition to institute an *inter partes* review of claims 1–15 and 31–45 of U.S. Patent No. 6,163,775 ("the '775 patent"). Paper 1 ("Pet."). We instituted trial for claims 1–15 and 31–45 on certain grounds of unpatentablility alleged in the Petition. Paper 14 ("Decision to Institute" or "Inst. Dec.").

After institution of trial, Patent Owner, Enfish, LLC ("Enfish"), filed a Patent Owner Response (Paper 30), along with a Declaration by Enfish's Declarant, Dr. H.V. Jagadish ("Jagadish Declaration"). On September 16, 2014, Enfish filed an unopposed motion to correct typographical errors in both papers and filed therewith the corrected Paper Owner Response ("PO Resp.," Paper 40) and the corrected Jagadish Declaration (Ex. 2007).[1]

Microsoft filed a Reply ("Reply"). Paper 35.

A consolidated hearing for IPR2013-00559, IPR2013-00560, IPR2013-00561, IPR2013-00562, and IPR2013-00563, each involving the same Petitioner and the same Patent Owner, was held on December 3, 2014. The transcript of the consolidated hearing has been entered into the record. Paper 64 ("Tr.").

We have jurisdiction under 35 U.S.C. § 6(c). This final written decision is issued pursuant to 35 U.S.C. § 318(a).

Microsoft has shown by a preponderance of the evidence that claims 31, 41, and 45 of the '775 patent are unpatentable. Microsoft has not

[1] For clarity, we refer to Enfish's corrected Patent Owner Response (Paper 40) and corrected Jagadish Declaration (Exhibit 2007), both filed on September 16, 2014.

sustained its burden of showing that claims 32–40 and 42–44 are unpatentable by a preponderance of the evidence. For the reasons discussed below, we are unable to reach a determination on the alleged grounds of unpatentability over prior art for claims 1–15. Accordingly, we terminate this proceeding with respect to claims 1–15 under 37 C.F.R. § 42.72.

B. Additional Proceedings

In addition to this petition, Microsoft has filed petitions challenging the patentability of claims 17–23 and 47–53, as well as claims 16, 24–30, 46, and 54–60 of the ’775 patent. *See* IPR2013-00560, and IPR2013-00561. Microsoft indicates that claims of the ’775 patent are presently asserted against Microsoft in *Enfish, LLC v. Microsoft Corporation, et al.*, Case No. CV12-7360 MRP (MRWx), in the Central District of California. Pet. 2. Microsoft further contends that a final judgment of invalidity for all the asserted claims in the lawsuit has been entered in the California case against Enfish. Ex. 1071.

Microsoft has also filed two petitions challenging the patentability of claims 1–60 of U.S. Patent 6,151,604 (“the ’604 patent”).[2] *See* IPR2013-00562; IPR2013-00563.

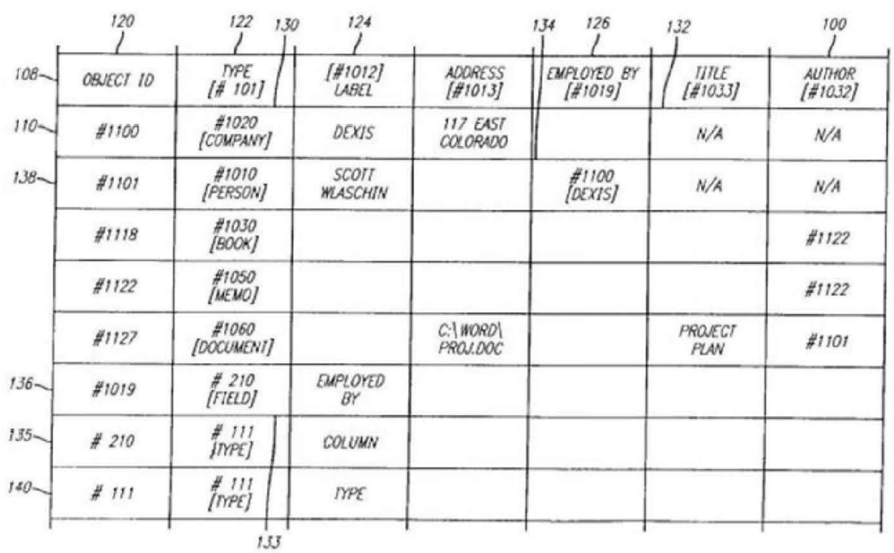
C. The ’775 Patent

The ’775 patent (Ex. 1001), titled “Method and Apparatus Configured According to a Logical Table Having Cell and Attributes Containing Address Segments,” generally relates to a system and method for data storage, manipulation and retrieval in a self-referential logical table of a

[2] The ’775 patent and the ’604 patents both issued from continuations of application No. 08/383,752, filed March 28, 1995, now U.S. Patent No. 5,729,730. Pet. 6.

database. This table is a logical structure, not a physical structure stored contiguously in the memory. *Id.* at 6:39–41.

Figure 3 of the '775 patent, reproduced below, illustrates a table structure of a logical table. *Id.* at 3:36–37.



As depicted by Figure 3 of the '775 patent, above, table 100 is defined by rows 108, 110, 138, 136, 135, and 140 and columns 120, 122, 124, and 126. Ex. 1001, Fig. 3. The intersection of a row and a column defines a cell in the table. *Id.* at, 6:48-49. Each column corresponds to an attribute spanning various records. *Id.* at 6:46. An attribute is a single class description, such as an employer, denoted in column 126 of Figure 3, for example, by the text "Employed By." *Id.* at 7:25-26.

Each row corresponds to a record spanning various attributes. Ex. 1001, 6:45-46. For example, row 110 corresponds to a company as shown in cell 130. *Id.* at 6:62–64.

D. Illustrative claims

Of the challenged claims, the independent claims are 1, 11, 15, 31, 41, and 45. Each of claims 2–10 depends, directly or indirectly, from claim 1. Each of claims 12–14 depends, directly or indirectly, from claim 11. Each of claims 32–40 depends, directly or indirectly, from claim 31. Each of claims 42–44 depends, directly or indirectly, from claim 41. Claim 31 illustrates the claimed subject matter and is reproduced below:

> 31. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of cells, each said cell having a first address segment and a second address segment;
>
> a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set having an object identification number (OID) to identify each said attribute set;
>
> a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein at least one of said logical rows has an OID equal to the OID of a corresponding one of said attribute set, and at least one of said records includes attribute set information defining each of said attribute sets.

E. Prior Art Relied Upon

Microsoft relies upon the following prior art references:

Chang et al., EP Publication No. 0 336 580 A2 (pub. Oct. 11, 1989) (“Chang,” Ex. 1003).

Dickerson et al., EP Publication No. 0 394 019 A2 (pub. Apr. 18, 1990) (“Dickerson,” Ex. 1003).

Crus et al., U.S. Patent No. 5,133,068 (issued Jul. 21, 1992) ("Crus '068," Ex. 1010).

Smith et al., U.S. Patent No. 5,181,162 (issued Jan. 19, 1993) ("Smith '162," Ex. 1008).

Goldberg et al., U.S. Patent No. 5,201,046 (issued Apr. 6, 1993) ("Goldberg '046," Ex. 1011).

Horn et al., U.S. Patent No. 5,226,158 (issued Jul. 6, 1993) ("Horn'158," Ex. 1005).

Smith et al., U.S. Patent No. 5,404,510 (issued Apr. 4, 1995) ("Smith '510," Ex. 1006).

Jenness, U.S. Patent No. 5,463,774 (issued Oct. 31, 1995) ("Jenness '774," Ex. 1013).

Anderson et al., U.S. Patent No. 5,463,724 (issued Oct. 31, 1995) ("Anderson '724," Ex. 1012).

Covey, U.S. Patent No. 5,745,755 (issued Apr. 28, 1998) ("Covey '755," Ex. 1014).

MARY E. S. LOOMIS, THE DATABASE BOOK (1987) ("Loomis," Ex. 1019).

KENNETH WEBB AND LORI LAFRENIERE, ORACLE DISTRUBUTED SYSTEMS-A C PROGRAMMER'S DEVELOPMENT GUIDE (1991) ("Webb," Ex. 1007).

MICROSOFT® VISUAL BASIC™ PROGRAMMING SYSTEM MANUAL FOR WINDOWS™ VERSION 3.0 (1993) ("Visual Basic," Ex. 1004).

IEEE, STANDARD DICTIONARY OF ELECTRICAL AND ELECTRONICS TERMS (4th Ed.) ("IEEE," Ex. 1018).

F. The Instituted Alleged Grounds of Unpatentability

Reference(s)	Basis	Claims challenged
Chang (Ex. 1003)	§ 102	1, 2, 31, 32, and 41
Chang and Horn (Ex. 1005)	§ 103	11
Chang and Webb (Ex. 1007)	§ 103	15 and 45
Chang and Smith '162 (Ex. 1008)	§ 103	3, 8, 33, and 38
Chang and Dickerson (Ex. 1009)	§ 103	4 and 34
Chang, Dickerson and Crus (Ex. 1010)	§ 103	5 and 35
Chang and Goldberg (Ex. 1011)	§ 103	6 and 36
Chang and Anderson (Ex.1012)	§ 103	7 and 37
Chang, Smith '162, and Jenness (Ex. 1013)	§ 103	9 and 39
Chang and Covey (Ex.1014)	§ 103	10 and 40
Chang, Horn, and Smith '162	§ 103	13 and 43
Chang, Horn, Smith '162, and Jenness	§ 103	14 and 44
Chang, Horn, and Anderson	§ 103	12 and 42

In support of the above-referenced grounds of unpatentability, Microsoft relies on the Declaration of Dr. Antony Hosking (Ex. 1022).

II. CLAIM CONSTRUCTION

A. Legal Standard

In an *inter partes* review, claim terms in an unexpired patent are

interpreted according to their broadest reasonable construction in light of the specification of the patent in which they appear. 37 C.F.R. § 42.100(b); *see also In re Cuozzo Speed Techs., LLC.*, No. 14-01301, slip op. at 16, 19 (Fed. Cir. Feb. 4, 2015) ("Congress implicitly adopted the broadest reasonable interpretation standard in enacting the AIA," and "the standard was properly adopted by PTO regulation."). Claim terms are given their ordinary and customary meaning as would be understood by a person of ordinary skill in the art at the time of the invention and in the context of the entire patent disclosure. *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). If the specification "reveal[s] a special definition given to a claim term by the patentee that differs from the meaning it would otherwise possess[,] . . . the inventor's lexicography governs." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1316 (Fed. Cir. 2005) (en banc) (citing *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002)). Also, we must be careful not to read a particular embodiment appearing in the written description into the claim, if the claim language is broader than the embodiment. *See In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993) ("[L]imitations are not to be read into the claims from the specification."). We apply this standard to the claims of the '775 patent.

B. Overview of the Parties' Positions

Microsoft contends that this case involves a computer-implemented invention. Pet. 7–10. Microsoft also contends that the challenged independent claims invoke means-plus-function claiming, but the '775 patent fails to disclose an algorithm, which is an issue under § 112, sixth paragraph, that cannot be addressed in this proceeding. Pet. 13. In the Decision to Institute, we invited Enfish to direct us to the specific portions of

the specification that clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. Inst. Dec. 12. For the reasons discussed below, we determine that Enfish in its response does not identify sufficient corresponding structure, as required under 35 U.S.C. § 112, sixth paragraph, for the "means for configuring said memory according to a logical table," recited in claims 1, 11, and 15.

With respect to the method claims 31–45, in the Decision to Institute, we provided constructions for "logical table," and "object identification number," which are shown in the table below. Inst. Dec. 12–13. We also determined that no express construction is needed for the following terms: "memory," "record," "attribute set," "at least one of said records includes attribute set information defining each of said attribute sets," and "defining the type of a different record," Inst. Dec. 15.

Claim Term or Phrase	Construction in the Decision to Institute
"logical table"	"[W]e construe the term "table" to mean: 'a structure of a database comprising rows and columns.'" Inst. Dec. 12. "We determine no express construction of 'logical' is needed for this decision." *Id.*
"object identification number"	"[W]e construe 'identification number' in light of the specification to mean: 'an array of bits that define.' For the purpose of this decision, an express construction of 'object' is not necessary." Inst. Dec. 13.

Enfish contends that our construction for "object identification number" is incomplete. PO Resp. 13. We evaluate Enfish's contention below. With the exception of the means-plus-function limitation, we discern no reason, based on the complete record now before us, to change our constructions in the Decision to Institute.

C. Analysis of the Parties' Claim Construction Positions

1. Means for configuring said memory according to a logical table

Independent claims 1, 11, and 15 include the limitation, "means for configuring said memory according to a logical table." In the Decision to Institute, we agreed with Microsoft that under the broadest reasonable interpretation, the function for the means for configuring is "configuring memory according to a logical table." Inst. Dec. 11. Additionally, we considered the corresponding structure for the recited function as including a general purpose computer. *Id.* at 11–12. Enfish does not challenge persuasively either of these determinations; however, Enfish identifies portions of the specification that Enfish contends provide algorithmic support for the recited function. PO Resp. 17–21. In particular, Enfish contends that the '775 patent discloses a four-step algorithm that is linked to the recited function of configuring memory according to a logical table. *Id.* at 18–20.

"[T]he corresponding structure for a § 112 ¶ 6 claim for a computer-implemented function is the algorithm disclosed in the specification." *Aristocrat Techs. Austl. Party Ltd. vs. Int'l Game Tech.*, 521 F.3d 1328, 1333 (Fed. Cir. 2008) (quoting *Harris Corp. v. Ericsson Inc.*, 417 F.3d 1241, 1249 (Fed. Cir. 2005)). Additionally, specific portions of the specification must clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. *See Medical Inst. & Diag. Corp. v. Elektra AB*, 344 F.3d 1205, 1211 (Fed. Cir. 2003). For the reasons set forth below, we conclude that the four steps and other '775 patent specification portions identified by Enfish do not describe an algorithm for

the recited function of "configuring memory according to a logical table." We further conclude that Enfish has not clearly linked or associated the recited function to the four steps or the portions of the '775 patent specification that Enfish identifies.

For algorithmic support, Enfish identifies disparate excerpts of the '775 patent specification, which do not link or associate clearly a computer program or algorithm to the function corresponding to the claimed means for configuring said memory according to a logical table. For example, the first step, "[c]reate, in a computer memory, a logical table" (PO Resp. 18) appears to be similar to the recited function of configuring memory according to a logical table, but the first step is not found in the '775 patent specification. Additionally, none of the portions of the '775 patent specification that Enfish cites for this first step provide an algorithm or computer program for performing the recited function of "configuring memory according to a logical table" or clearly link or associate any algorithm or program to this recited function. PO Resp. 18 (citing Ex. 1001, Abs., 2:59–63, 6:38–46, Figs. 1, 3, 9). These portions describe that an already-formed table has rows and columns, without describing how memory is configured to create a logical table having rows and columns. Ex. 1001, 2:59–63. One portion states that memories "need not store" the table contiguously, but fails to describe an algorithm or computer program for configuring memory such that a logical table is not stored contiguously. *Id.* at 6:38–46.

Enfish's three remaining steps and the other '775 patent specification portions identified by Enfish fail to remedy these deficiencies. The second step is: "[a]ssign each row and column an object identification number

(OID) that, when stored as data, can act as a pointer to the associated row or column." PO Resp. 18–19. One of the portions of the '775 patent specification cited by Enfish for the second step indicates "the system must generate a unique OID when columns and rows are formed." Ex. 1001, 8:15–16. The remainder of the identified portions relate to assigning an OID or indicate that an OID "may be used" as a pointer, without describing an algorithm for forming columns and rows of a table or showing how assigning an OID relates to steps for forming columns and rows. Ex. 1001, Abstract, 2:60–61, 6:50–57, 7:1–2, 8:18–60, Figs. 3, 4. Additionally, the identified portions discuss assigning a numeric value to an OID in the form of a bit array, but fail to describe how to configure memory such that an OID may be used as a pointer. *Id.*

The third step is "[f]or each column, store information about that column in one or more rows of the logical table." The portions of the '775 patent specification cited by Patent Owner for the third step (PO Resp. 25 (citing Ex. 1001, Abstract, 2:66–3:2, 6:47–48, 7:25–31, Fig. 3)) indicate that a table has a row that corresponds to columns, and the row contains information about the column, such as a header row, which is a generic feature of an already-formed table. These '775 patent specification portions identified by Enfish (*id.*) describe the table having such corresponding rows and columns, however, these excerpts do not describe how to form a table with this feature or link or associate this feature to the recited function.

The fourth step, i.e., storing and accessing data in cells (PO Resp. 19–20), is performed in an already-configured table and, therefore, occurs after the recited function of configuring a table has occurred. Nonetheless, the '775 patent specification portions identified by Enfish (*id.* (citing Ex. 1001,

Abstract, 2:63–66, 6:48–49, 6:61–62, 7:9–10, 7:23–24, 11:52–62, Fig. 10)) suffer from the same deficiencies noted above.

We conclude that the four steps and other ’775 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table. The portions of the ’775 patent specification identified by Enfish describe an already-formed table having generic characteristics, such as columns, rows, identifiers, and a header row. *See, e.g.*, Ex. 1001, Fig. 3. The description, however, does not disclose any algorithm or computer program for forming this table. Additionally, the description identified by Enfish is not linked or associated clearly with the recited function of “configuring memory according to a logical table.”

Enfish also relies on Dr. Jagadish’s Declaration to show that the ’775 patent specification provides an algorithm and clearly links the algorithm to the recited function. PO Resp. 18–20 (citing Ex. 2007 ¶¶ 68–76). The Jagadish Declaration, however, relies on similar portions of the ’775 patent specification cited in Patent Owner’s Response. For the reasons given, the Jagadish Declaration does not support Enfish’s assertion. Dr. Jagadish further asserts, “[i]t is also my opinion that one of ordinary skill in the art would understand how to implement those algorithm steps using techniques and resources that were available at the time the ’775 Patent was filed.” Ex. 2007 ¶ 69. Enfish, however, cannot rely on the knowledge of one skilled in the art to address the deficiencies noted above. *See Function Media, LLC v. Google Inc.*, 708 F.3d 1310, 1319 (Fed. Cir. 2013) (“Having failed to provide any disclosure of the structure . . . FM cannot rely on the knowledge of one skilled in the art to fill in the gaps.”)

Because we conclude that the four steps and other ’775 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table, we terminate this proceeding with respect to the claims that recite this means-plus-function limitation. As explained in *BlackBerry Corporation v. Mobile Media Ideas, LLC*, Case IPR2013-00036 (PTAB Mar. 7, 2014) (Paper 65), the specification must disclose enough of a specific algorithm to provide the necessary structure under § 112, sixth paragraph. In the circumstance when the specification of the challenged patent lacks sufficient disclosure of structure under 35 U.S.C. § 112, sixth paragraph, the scope of the claims cannot be determined without speculation and, consequently, the differences between the claimed invention and the prior art cannot be ascertained. *Id.* For the reasons given, we determine that independent claims 1, 11, and 15 are not amenable to construction and, thus, we terminate this proceeding with respect to claims 1, 11, and 15 under 37 C.F.R. § 42.72. Because claims 2–10 depend, directly or indirectly, from claim 1, and claims 12–14 depend, directly or indirectly, from claim 11 we also terminate this proceeding with respect to these claims under 37 C.F.R. § 42.72.

2. *Means for configuring said memory according to a logical table*

Because we determine that this proceeding should be terminated as to claims 1–15 for the reasons discussed above, we need not construe other means-plus-function terms appearing in those claims for the purposes of this Decision.

3. Object Identification Number (OID)

Independent claims 1, 11, and 15 recite "object identification number (OID)" and "OID." In the Decision to Institute, we construed "object identification number" in light of the specification to mean: "an array of bits that define an object." Inst. Dec. 13 (citing Ex. 1001, 3:38–40, 8:43–45).

Enfish "agrees in principle" with our construction, but contends that the construction is incomplete. PO Resp. 13. Enfish contends that "OID" should be defined further as "a unique, immutable, and system-generated value that identifies an object." *Id.* Microsoft contends that Enfish seeks to read into the claims extraneous limitations that are unsupported by either intrinsic or extrinsic evidence. Pet. Reply 2. Microsoft additionally contends that Enfish's proposed construction does not provide an appropriate context for the additional limitations. *See, e.g.*, Pet. Reply 3.

At the heart of Enfish's contention is an assertion that an OID will not function properly if it is not unique, immutable, and system generated. PO Resp. 13–17. For example, Enfish contends that an OID must be a unique value because "if an OID were not unique the database would be non-functional." PO Resp. 14. Enfish asserts that "The invention assigns the unique OIDs to every row and every column: 'Each row is assigned a unique object identification number (OID) stored in column 120 and each column also is assigned a unique OID, indicated in brackets and stored in row 108.'" PO Resp. 14–15 (citing Ex. 1001, 6:50-52).

Microsoft contends that Enfish's "unique" OID requirement would conflict with the patent, which describes a row in a table with an OID that has the same value as the OID of a column in the same table. Pet. Reply 3 (citing Ex. 1001, Fig. 3). Our examination of the '775 patent indicates that a

column can have the same OID as a row, in fact, this is the particular aspect of the table being self-referential. Ex. 1001, 2:66–3:3 ("To enhance searching and to provide for synchronization between columns, columns are entered as rows in the table and the record corresponding to a column contains various information about the column. This renders the table self referential"). Figure 3 of the '775 patent discloses row 136 (corresponding to column 126) having OID #1019, and column 126 also having OID #1019. To the extent such an OID is unique, we are persuaded that it is unique "to every row and column" as Enfish asserts. PO Resp. 14–15.

Microsoft also contends that "immutable" should not be imported into the construction of OID because the term is absent from the specification, claims, and file history, and the specification is devoid of disclosure relating to immutability of an OID. Pet. Reply 3–5. Microsoft alleges that PO relies solely on extrinsic support for OID being "immutable. *Id.* at 4.

We determine that adopting Enfish's proposed construction without further context would create ambiguity. Limitations are not to be read into the claims from embodiments in the specification. *See In re Van Geuns*, 988 F.2d at 1184. Furthermore, we agree with Microsoft that Enfish relies on insufficient extrinsic evidence for support, for example, that an OID is "immutable." We, therefore, determine that the construction of "OID" in the Decision to Institute should not be changed.

III. ANALYSIS

For the reasons given below, Microsoft has shown, by a preponderance of the evidence, that each of claims 31 and 41 is unpatentable under 35 U.S.C. § 102 as anticipated by Chang and that claim 45 is unpatentable under 35 U.S.C. § 103 as obvious over Chang and Webb.

Also, we determine that Microsoft has not shown by a preponderance of the evidence that claims 32–40 and 42–44 are obvious over Chang as combined with one or more of Smith '162, Dickerson, Crus, Goldberg, Anderson, Jenness, Covey, and Horn.

A. Anticipation by Chang

We have reviewed Microsoft's anticipation argument and supporting evidence, including Chang's disclosure and the detailed explanation appearing on pages 21–40 of the Petition. Microsoft's explanation persuasively reads all limitations of claims 31 and 41, but not claim 32, onto the disclosure of Chang. Despite the counter-arguments in Enfish's Patent Owner Response, and the evidence cited therein, which we also have considered, Microsoft has shown, by a preponderance of the evidence, that each of claims 31 and 41 are unpatentable under 35 U.S.C. § 102(b) as anticipated by Chang. *See* 35 U.S.C. § 316(e).

1. Chang

Chang discloses a relational database including EMPLOYEE TABLE 10 ("EMP") in Figure 1, which includes a number of records, each record (row) containing specific data relating to a certain employee. Each record in EMPLOYEE TABLE 10 includes the employee number, name, salary, and department number aligned in columns 1–4 respectively. Ex. 1003, 5:26-33. The relational database also includes other tables, i.e., system catalogs, as shown in Figures 2–4. The system catalogs contain further information about the tables in the database, for example SYS.TABLES 12 in Figure 2 contains a record for EMPLOYEE TABLE 10 including the table name, and the creator in columns 1 and 2. Chang states that SYS.TABLES 12 may also contain information about a different table, such as an employee

location table (not shown) including the employee number and location. *Id.* at 5:40–45. Chang explains that:

> Due to common elements in these various tables relational information may be derived by using this commonality. For example, because the employee number is common to the aforementioned employee table and site location table, this piece of information may be used to relate a particular employee's name from the first table to the location where the employee works derived from the second table.

Id. at 5:45–53.

Chang also discloses storing definitional information relating to column attributes in a single record (row) of a table so that "[a]ccessing the row corresponding to a particular object returns a description of all of the attributes of the object's (i.e. the column) component objects, as well as information describing the object itself." *Id.* at 3:58–4:3. More specifically, Chang explains that the table shown in Figure 3 as reproduced below, SYS.COLUMNS catalog 14, contains records relating to the database structure, i.e. database objects and component objects, of the EMPLOYEE TABLE 10.

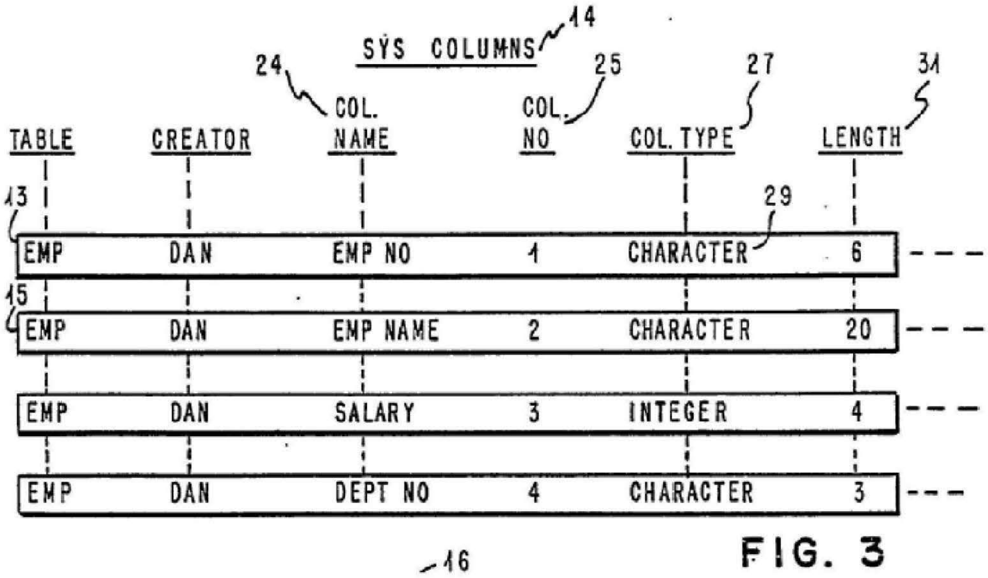


FIG. 3

Figure 3, above, reveals that SYS.COLUMNS catalog 14 contains specific information about the attributes of each of the columns in the EMPLOYEE TABLE 10. *Id.* at 6:25-39. Chang describes that:

> each record in the SYS.COLUMNS catalog 14 shown therein contains a number of fields describing attributes of a correlative one of the columns in the employee table 10…for example, EMP in the "Table" column as the first field thereof indicates that the data to the right is further information or attributes about a column appearing in the employee table.

Id. at 6:40–44. Each record (row) in the SYS.COLUMNS catalog 14 includes a field ("COL.NO" 25) having the column number relating to the corresponding column in the EMPLOYEE TABLE 10, which is defined by that particular row. For instance, from the SYS.COLUMNS catalog 14 in Figure 3, the value "1" in COL.NO 25 correlates this record with column 1 in the EMPLOYEE TABLE 10 shown in Figure 1. *Id.* at 6:50–7:10.

2. Claims 31 and 41

We have reviewed Microsoft's anticipation argument and supporting evidence which relates persuasively each element of claims 31and 41 to the disclosure of Chang. Pet. 22–40.

Microsoft asserts generally that Chang discloses a relational database and that it is well known that databases are used to store and retrieve data. Pet. 23 (citing Ex. 1003, 1:3–4, Abst., Ex. 1022 ¶ 170). Microsoft contends that Chang describes implementing the database on a computer system having memory, a central processing unit, and a display. *Id.* (citing Ex. 1003, Fig. 11). Microsoft further asserts that Chang discloses a method for configuring the memory according to a logical table that organizes data into rows and columns, and that the intersection of the rows and tables, by combination of the row/record i.d. and column value, identifies a cell. *Id.* at 24 (citing Ex. 1003, 13:57–14:10, Figs. 2–5, 11, Ex. 1022 ¶¶ 173–175, 182–184). Microsoft alleges that the cells in each column (attribute set) have the same second address segment, such as the same column number, to identify each cell. *Id.* at 28 (citing Ex. 1022, Hosking Decl. at ¶¶ 176–178). The OID's, Microsoft contends, are disclosed at least by "[t]he column number and/or ordinal number [] values that identify an object. Here, that object is a column." *Id.* at 29. Microsoft's arguments here relate to portions [A], [B], [C], and [D] of claim 31 for example, reproduced below. We are persuaded that Chang discloses these initial limitations of claim 31 because, as Microsoft's evidence shows, Chang describes a relational database having objects such as tables, and columns for storing data in a determinable way. Ex. 1003, 1:1–14. Chang further explains how various data entry, search,

and manipulation is carried out with respect to such objects. *Id.* at 27–37, Also, Chang discloses, in accordance with our claim construction, that the database tables include column identifiers, such as column numbers, and columns names, comporting with the proper interpretation of an OID, as "an array of bits that define" a column. *Id.* at 5:41–45.

Turning to the remaining limitations in claims 31 and 41, referenced as [E], [F1], and [F2], Enfish makes three specific arguments regarding why Chang does not disclose all the limitations of independent claims 31, and 41. PO Resp. 22. For purposes of clarity, we reproduce below method claim 31, including the additional reference elements ([E] and [F1]) referred to by Enfish.

> 31. [A] A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:
>
> [B] configuring said memory according to a logical table, said logical table including:
>
> [C] a plurality of cells, each said cell having a first address segment and a second address segment;
>
> [D] a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and
>
> [E] a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, [F1] wherein at least one of said records has an OID equal to the OID of a corresponding one of said attribute sets, and [F2] at least one of said records includes attribute set information defining each of said attribute sets.

Enfish asserts first that no one, single table in Chang includes all the limitations of these independent claims. *Id.* Second, Enfish contends that

neither of Chang's two tables (SYS.TABLES, SYS.COLUMNS), disclose the row OID limitation [E], and third, neither table discloses the limitation [F1] requiring a row OID to be equal to a column OID. *Id.* Enfish's contentions are based on its proposed construction of OID, which we decline to adopt for the reasons given above. Nonetheless, we disagree with Enfish's first assertion because, while we are persuaded by Enfish's position that the claims require all the elements to be found in a single table, Microsoft provided persuasive evidence that Chang's SYS.COLUMNS, at least inherently, if not expressly, includes a record row that defines each of the columns that appear in the SYS.COLUMNS table itself. Pet. 37. To anticipate a patent claim under 35 U.S.C. § 102, "a single prior art reference must expressly or inherently disclose each claim limitation." *Finisar Corp. v. DirecTV Group, Inc.*, 523 F.3d 1323, 1334 (Fed. Cir. 2008). In other words, SYS.COLUMNS, although shown by way of example in Figure 3 of Chang for EMPLOYEE TABLE ("EMP"), will, in the Table field also have a correlating SYS.COLUMNS row, having a creator (CREATOR) field, column name (COL NAME) field, e.g. column name, column number, and a column number (COL.NO) field including the column number 1, 2, 3 . . .etc., defining the SYS.COLUMNS columns. *See* Ex. 1003 at 6:49–7:19. In light of our interpretation of OID set forth above in II.C.3., Microsoft persuasively argues that "[a] row holding a column definition will of course hold an OID that is the same as the OID of the defined column." Pet. 37, *see also* Reply 10–11.

Enfish's second position, specifically that the OID's identified by Microsoft do not satisfy the "record including" part of the claim limitation [E] ("a plurality of records, each said record including a series of cells

having the same first address segment, each said record including an OID to identify each said record,") "because no cell in any row of Annotated FIG. 3[] contains all these values together," is also not persuasive. PO Resp. 31. Enfish did not advance any claim construction for the term "record including." The limitation reads in context "each said record including," not "*a cell of* each said record including." Indeed, the evidence asserted by Enfish, specifically Dr. Hosking's deposition testimony, indicates that an OID *can* be stored in a cell, not that it *must* be stored in a cell, in order to identify the row. *Id.* at 30, Ex. 2003, 183:12–184:17. Accordingly, we give little weight to Enfish's interpretation of Dr. Hosking's testimony. We are not apprised by Enfish of any persuasive evidence from the specification of the '775 patent, or otherwise, that the claims require that an OID *must* be stored in a cell of either a row or column.

In its third argument, Enfish maintains that element [F1] ("wherein at least one of said records has an OID equal to the OID of a corresponding one of said attribute sets") is not meet by Chang because none of the OID values considered by Microsoft (record i.d., column number, column name, a combination of the table name, creator name, and/or column name or number) "satisfy all three of the requirements of OIDs—uniqueness, immutability and system generated." PO Resp. 30. Enfish's claim interpretation does not address the limitation in terms of the claim construction for OID provided in these proceedings. We determine that the OID values, alone and in combination, identified by Microsoft satisfy our construction for an OID as "an array of bits that define." Each row in SYS.COLUMNS, shown in Figure 3 of Chang, includes both the table name (column 1), as well as column number (column 4). Therefore, for at least

the table name and column number values, included in the row defining a particular column, SYS.COLUMNS satisfies the [F1] limitation "wherein at least one of said records has an OID equal to the OID of a corresponding one of said attribute sets," as recited in claim 31.

Enfish does not address the additional limitation [G] in claim 41 "searching said table for said pointer" in either the Preliminary Response or Patent Owner's Response. *See* Prelim. Resp. 21–31, PO Resp. 22–47. Microsoft explains how this limitation of claim 41 is met by Chang. Pet. 38–39 (citing Ex. 1003, Fig. 5, Ex. 1022 ¶¶ 204–207, 297–282. Chang discloses searching a relational database in which columns are amenable to being searched by disclosing that certain columns are indexed. *See* Ex. 1003 at Figure 5. Microsoft asserts that the ordinary use of such an index is searching the index to return a particular row or rows, from which the OID, acting as a pointer, described for limitation [G] can be determined. Pet. 39 (citing Ex. 1022, Hosking Decl. at ¶¶ 204–207, 275–277). We are persuaded by this explanation.

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claims 31 and 41 of the '775 patent are anticipated by Chang.

3. Claim 32

Enfish further argues that the elements of claim 32 are not found in a single table, but are only found in SYS.INDEXES catalog of Figure 4, which is not a part of SYS.TABLES (Figure 2) or SYS.COLUMNS (Figure 3). PO Resp. 47–48. Microsoft does not address this argument in their Reply. As discussed above, we are persuaded that claims 31 and 32 require all the claim limitations following "said logical table including" to be found in the

same logical table. The Federal Circuit has repeatedly emphasized that the indefinite article "a" carries the meaning of "one or more" in open-ended claims containing the transitional phrase "comprising." *Baldwin Graphic Sys., Inc. v. Siebert, Inc.*, 512 F.3d 1338, 1342 (Fed. Cir. 2008). Claim 31 recites "a logical table," and while the claim is not restricted to a database with one table, we determine that the recitations following "said logical table including" in claim 31 must be found in a single table regardless of the number of tables in the database. Claim 32 further limits the "attribute set information" recited as an element of the logical table in claim 31.

Our review of the specification of the '775 patent does not reveal any evidence, or embodiment, nor does Microsoft point us to any evidence, indicating that the elements of Enfish's claimed "logical table" were intended to be in separate, different tables. *See* Ex. 1001 Abst., 1:36–55, 3:13–27. The specification of the '775 patent consistently refers to table 100 in the context of a single table, including, in an embodiment used in a word processing application. *Id.* at 17:16–18, Figs 3, 9, 10, 13. ("The database of the present invention includes a novel Structured Word Processor that may be used in conjunction with the table 100.")

The evidence provided by Microsoft with respect to the packed description in SYS.TABLES storing column information, including primary key column definition information from the SYS.COLUMNS table is not persuasive because it does not explain how OID determination, by text entry, as recited in claim 32 would be conducted on such stored information and definitions in these particular tables, or even that SYS.TABLES or SYS.COLUMNS are indexed to provide such a search function. Pet. 39–40. Dr. Hosking states that it is SYS.INDEX of Figure 4 that is necessary for

Chang to determine what "columns are amenable to being searched." Ex. 1003 ¶ 200. Enfish argues persuasively that the only table Chang discloses for index searching is SYS.INDEX shown in Figure 4. PO Resp. 48 (citing Ex. 1003, 7:27–29, Ex. 2003 at 193:3-6).

For the foregoing reasons, Microsoft has not shown, by a preponderance of the evidence, that claim 32 of the '775 patent is anticipated by Chang.

B. Obviousness over Chang and Webb

We have reviewed Microsoft's obviousness arguments and supporting evidence, including Chang and Webb's disclosure and the detailed explanation appearing on pages 46–48 of the Petition. Microsoft's explanation persuasively reads all elements of claim 45 onto the disclosure of Chang. For the reasons given below, Microsoft has shown, by a preponderance of the evidence, that claim 45 is unpatentable as obvious over the combination of Chang and Webb.

1. Webb

Webb describes a distributed application which runs on a local computer and accesses data stored on a central computer. Ex. 1007, 14. The central computer makes the same data accessible to multiple users and application programs. *Id.* Webb describes itself as a "[g]*uide* . . . for C programmers who want to embed Structured Query Language (SQL) statements into their programs for access to local or remote (distributed) databases." *Id.* Webb also discloses table 3-5 including several rows, PRODUCT_ID 102 through 108, each row having pointer 101, that points to PARENT_PRODUCT_ID 101 which is a separate record/row. *See* Ex. 1007 at 47.

2. Claim 45

Enfish argues that Microsoft does not provide sufficient rationale for combining Chang and Webb. PO Resp. 58–59. Enfish specifically argues "[n]either the Petitioner nor Dr. Hosking, however, have even discussed such a combination, let alone demonstrated that such a combination is possible." *Id.* at 58 (citing Prelim. Resp, 53–56). We are not persuaded by this argument. In contrast to this statement, Microsoft explained that Webb provides a flexibility desired by relational database designers and relates to a relational database system and specific database structure, which includes SQL (Structured Query Language) statements, and discloses table 3-5 where at least one of the records/rows contains a cell that contains a pointer to a different row." Pet 46–47 (citing Ex. 1007 at 47, Ex. 1022 ¶¶ 287–288.).

We are persuaded that Microsoft has shown a sufficiently articulated reason with rational underpinning to support obviousness. Pet. 47–48 (citing Ex. 1022 ¶¶ 289–297). Microsoft explained that one of ordinary skill in the art would have understood the SQL language and would have combined Webb's database structures with Chang's system in order to "consolidate[e] system catalog information to reduce access requirements within a relational database." *Id.* at 48. The predictable use of familiar prior art elements according to their established functions renders the recited invention obvious. *See KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 417 (2007).

Microsoft explains how the initial limitations [A]–[E], and limitation [H3] of claim 45 are met by Chang for the same reasons as claims 31 and 41 discussed above in section III.C.2. Pet. 46. Microsoft additionally explains how the remaining elements of claim 45, listed below as [H1], [H2] as referred to by Microsoft and Enfish, are met by Webb. Pet. 46–48 (citing

Ex. 1007, xiii, 47, Ex. 1022 ¶¶ 287–297.

> [H1] at least one of said plurality of records contains a cell having a pointer to a different record
>
> [H2] at least one of said plurality of records includes information defining the type of a different record; and
>
> [H3] searching said table for said pointer (claim 45).

Microsoft contends that Webb discloses table 3-5 having several rows that point to the parent product ID of a different record/row. Pet. 47 (citing Ex. 1007, 47, Ex. 1022 ¶¶ 287–288). Microsoft further argues "that rows with PRODUCT_ID of 102 through 108 are all of the same type, i.e. category, defined by the row for PRODUCT_ID 101, as indicated in the PARENT_PRODUCT_ID column. *See* Ex. 1007 at 47. We are persuaded by this explanation. For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 45 of the '775 patent is unpatentable as obvious over the combination of Chang in view of Webb.

C. Obviousness over Chang and Smith '162, or over Chang, Horn, and Smith '162, or over Chang, Smith '162, and Jenness, or over Chang, Horn, Smith '162, and Jenness

We have reviewed Microsoft's obviousness arguments and supporting evidence, including Chang and Smith '162's disclosure and the detailed explanation appearing on pages 48–50 of the Petition. For the reasons given below, Microsoft has not shown, by a preponderance of the evidence, that claims 33, 38, 39, 43, and 44 are unpatentable as obvious over the combination of Chang and Smith '162.

1. Smith '162

Smith '162 describes an object-oriented database system where

various documents are organized and represented as collections of logically related documents, i.e. documents of a similar type, or "objects." Ex. 1008, Abst. These "objects" containing the document collections can be "generically referred to as 'folders,'" and the folders represented as "objects." *Id.* at 3:29–35. The underlying database management system has an organizational table structure with rows and columns, where columns can specify objects, including attributes of an object, representing a folder. *See Id.* at 9:66–68, 10:31–34, Fig. 1.

2. *Claims 33 and 38*

Enfish asserts that Microsoft fails to provide a sufficient rationale for combining Chang and Smith '162. Prelim. Resp. 41. Enfish contends that Microsoft's only motivation for the combination is that "Chang and Smith '162 are closely related and address the same technical issues." Prelim. Resp. 41 (citing Pet. 49:10–12). Enfish argues that the bare assertion by Microsoft and Dr. Hoskings, that two references are within a similar field of endeavor, is insufficient to support a determination of obviousness. *Id.* (citing Ex.1022 ¶ 307).

We agree with Enfish that Microsoft has not met its burden to show that a person of ordinary skill in the art would have modified Chang based on the disclosure of Smith '162 resulting in the subject matter of the challenged claims. Microsoft asserts that "a person of ordinary skill in the art would have been motivated to combine Chang with Smith '162 because Chang and Smith '162 are closely related and address the same technical issues." Pet. 49 (citing Ex. 1022 ¶¶ 298–308). In support of this statement, Microsoft contends that Chang discloses a "file i.d." for each record, "that contains groups of records of a similar type," but does not explicitly disclose

this FID column object as a folder. *Id.* (citing Ex. 1008, 10:31–34). Petitioner further contends that Smith '162 discloses folder objects which contain objects, such as documents, of a similar type. *Id.* (citing Ex. 1008 9:66–69, Fig. 1).

Our review of Chang reveals simply that "file i.d." represents a file location "from which additional information on the table may be derived." Ex. 1003, 6:5–10. Microsoft's restatement of Chang's disclosure extrapolates "groups of records of a similar type" from "additional information," without sufficient explanation or evidence for this factual leap. *See* Pet. 49. A claim is unpatentable for obviousness under 35 U.S.C. § 103(a) if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. *See KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). A patent claim composed of several elements, however, is not proved obvious merely by demonstrating that each of its elements was known, independently, in the prior art. *Id.* at 418. In analyzing the obviousness of a combination of prior art elements, it can be important to identify a reason that would have prompted one of skill in the art to combine the elements in the way the claimed invention does. *Id.*

Microsoft's argument does not explain why one of skill in the art would look to Smith '162 for a "folder type record." Microsoft has not articulated any reason, or presented rational evidentiary underpinnings, explaining why Chang's "file i.d." was, or should have been, understood as a "folder object" as disclosed by Smith '162, or why one of skill in the art at the time of invention of the subject matter of the '775 patent would have

looked to combine Smith '162's "folder objects" with Chang "file i.d."

We conclude that Microsoft has not shown by a preponderance of the evidence that claims 33 and 38 would have been obvious over Chang and Smith '162.

3. Claims 39, 43, and 44

For claim 39, Microsoft relies on the combination of Chang, Smith '162, and Jenness. Pet. 54–55. Microsoft contends that Jenness teaches a pointer to the folder allegedly disclosed by Smith '162. *Id.* (citing Ex. 1013 5:62-63). As explained above with regard to claims 33 and 38, we conclude that Microsoft has not articulated any reason, or presented rational evidentiary underpinnings, explaining why Chang would have been combined with Smith. Thus, claim 39, which depends from claim 38, is not obvious for the same reason as claim 38. To meet the limitations of claim 43, Microsoft relies upon the combination of Chang, Horn, and Smith '162 and states that "it would have been obvious to combine Chang and Smith '162, as described for claim [] 38, with the further teachings of Horn. Pet. 57. Chang and Smith '162, however, cannot be combined to meet claim 43 for the same reasons discussed above in regard to claim 38. Claim 44 depends from claim 43, thus for the same reasons discussed above, Chang and Smith '162 cannot be combined to meet claim 44. Accordingly, we conclude that Microsoft has not shown by a preponderance of the evidence that claims 39, 43, and 44 would have been obvious over Chang, Smith '162, and Jenness, and/or Horn.

D. Obviousness over Chang and Dickerson, or over Chang, Dickerson, and Crus

For the reasons given below, after consideration of Microsoft's

Petition and Enfish's PO Response, and the evidence cited therein, Microsoft has not shown by a preponderance of the evidence that claims 34 and 35 are unpatentable as obvious over the combination of Chang and Dickerson, or Chang, Dickerson, and Crus.

1. Dickerson

Dickerson discloses a database system residing on a host, or central computer, and users can access the database from terminals connected to the central computer. Ex. 1009, 1:25–45, Fig. 1. Dickerson also teaches a contingency system having the same database at two different sites, with both sites being accessible by users from other terminals, where both databases are updated to maintain the databases identical. *Id.* at 11:5–20. Dickerson states "[w]henever a resource such as a part of a database is accessed during a sync interval, a lock will be placed on both copies of that resource (one copy at each site) . . . the two databases will always remain identical for practical purposes as the locks on the data at the second site will not be released until its commit process is completed, so that no terminal will ever be able to obtain differing information from the two databases." *Id.* at 11:9–20.

2. Claim 34

Microsoft argues that Chang discloses all the limitations of base claim 31, but not the reciprocal synchronization of two attribute sets recited in claim 34. Pet. 50. More particularly, Microsoft asserts that Dickerson discloses the limitation in claim 34 "*wherein said attribute set information defines one of said attribute sets to contain information for synchronizing two attribute sets reciprocally.*" *Id.* Microsoft argues that Dickerson discloses "mirroring," that teaches "reciprocal synchronization of two

attribute sets," where Dickerson discloses modification of an attribute in one database replicated in databases at other sites. *Id.* at 50–51 (citing Ex. 1009 at 10:56-11:4, Figure 8). In support of this ground, Microsoft relies on the Hosking Declaration (Ex. 1022 ¶¶ 248, 309–320), explaining how each limitation is disclosed in Chang and Dickerson. Pet. 51.

Enfish contends that Microsoft's arguments are based on "mirroring" that "occurs across separate databases as opposed to across two columns within the same table." PO Resp. 50. Enfish makes two specific arguments with respect to "mirroring" and the asserted combination. *Id.* Enfish first contends that Petitioner has failed to explain, or provide evidence, why one of skill in the art would combine these references to "mirror" columns within the same logical table in the first place. *Id.* Enfish asserts that, on the contrary, one of skill in the art would understand that "mirroring" columns across different databases defeats the purpose of a single system catalog such as SYS.TABLES and SYS.COLUMNS in Chang. *Id.* Also, Enfish contends that "mirroring" occurs across separate databases and is not the same function as "synchronizing two attribute sets reciprocally" as recited in claim 34. *Id.*

Dickerson teaches replication of data across separate databases at different sites by synchronization:

> Whenever a resource such as a part of a database is accessed during a sync interval, a lock will be placed on both copies of that resource (one copy at each site). A communications function local to the transaction manager will participate in the commit process with that transaction manager so as to guarantee that any update information is transmitted out of that site by the time that commit takes place. The transmitted update information is used to update the database at the other site so as to maintain the two databases identical.

Ex. 1009, 11:9–20. Dickerson states that mirroring "is an arrangement in which databases at different sites hold the same data, as in a contingency system, but with the databases at both sites being available for access by terminals." *Id.* at 10:58–11:4. The evidence in Dickerson relating to mirroring data in separate databases reveals a discrete sync interval permitting the transfer of the same data, in the same attribute set, between separate databases and separate tables. *Id.* at 5:56–6:5. This disclosure, however, does not describe how to synchronize two attribute sets in the same table by a specific synchronizing attribute set, also in the same table, as called for in claim 34.

We give little weight to Dr. Hosking's testimony with respect to the limitations in claim 34 because he does not provide an explanation or evidence as to why Dickerson's synchronization of data in the same attribute set across separate databases teaches reciprocal synchronization of attribute sets in the same table. Dr. Hosking states that a person of skill in the art would be motivated to combine Chang and "the synchronization concept as taught in Dickerson in the case that similar elements were to be reciprocally synchronized." Ex. 1022 ¶ 248. The premise of this argument is the assumption that Dickerson's synchronization across separate databases is the same as synchronizing two attribute sets reciprocally in the same database table. Dr. Hosking does not provide support for this assumption. This argument, therefore, does not sufficiently explain why mirroring data across separate databases leads to data synchronization between attribute sets in the same table. Dr. Hosking's testimony does not explain why, or how, one of ordinary skill in the art would use the database management disclosed by

Chang in cooperation with the linked database elements between separate databases to achieve reciprocal synchronization between attribute sets in the same table. *See id.* ¶¶ 310–320.

We conclude that Microsoft has not shown by a preponderance of the evidence that claim 34 would have been obvious over Chang and Dickerson.

3. Claim 35

For claim 35, which depends from claim 34, Microsoft relies on the combination of Chang, Dickerson, and Crus. Pet. 51–52. Microsoft contends that Crus teaches "reciprocal pointers to said two attribute sets" as required by claim 35. Chang and Dickerson cannot be combined to meet claim 35 for the same reasons as set forth above with respect to claim 34. Accordingly, we conclude that Microsoft has not shown by a preponderance of the evidence that claim 35 would have been obvious over Chang, Dickerson, and Crus.

E. Obviousness over Chang and Goldberg

For the reasons given below, after consideration of Microsoft's Petition and Enfish's PO Response, and the evidence cited therein, Microsoft has not shown, by a preponderance of the evidence, that claim 36 is unpatentable as obvious over the combination of Chang and Goldberg.

1. Goldberg

Goldberg teaches a database management system for a relational database which facilitates storage and retrieval of what are known as "directed graphs." Ex. 1011, 1:7–10, Fig. 1. Goldberg also discloses that a column in a table will have a reference data type, and that entries in that column "are pointers to rows in a specified table." *Id.* at Abst. Goldberg further explains that the directed graph structure is stored as a record/row in

a table, "with references corresponding to interconnections between records being stored in reference data type columns." *Id.*

2. *Claim 36*

Microsoft relies on Dr. Hosking's Declaration to explain how each limitation of claim 36 is disclosed in Chang and Goldberg. Pet. 52-53 (citing Ex. 1022 ¶¶ 335-339). In the Petition, Microsoft addresses only the second limitation in claim 36, not the first limitation. *Id.* In Dr. Hosking's Declaration, however, he states that Chang discloses, "at least one of said plurality of records includes information defining the type of a different record," addressing the first limitation recited in claim 36, With respect to this first limitation, Dr. Hosking testifies that:

> 336. As previously stated, Chang discloses a Packed Description, PD, in a row that contains information about columns that are represented by rows. *See* Claim 1[F] and 31[F] Anticipation by Chang Discussion. The PD includes information that defines the column type. *Id.* The column is represented as a row in the SYS.COLUMNS table. *Id.* Accordingly, the PD disclosed by Chang includes information that defines the type of a different logical row.

Ex. 1022 ¶ 336.

Enfish argues that Dr. Hosking's rationale is flawed in that it relies upon the Packed Description in SYS.TABLES table, and the column defining the Packed Description in SYS.COLUMNS table, and therefore does not meet the single table requirement of base claim 31. PO Resp. 51–52. Enfish contends that the correct reading of claims 31 and 36, "[t]he 'at least one of said plurality of records' and the 'different record' are recited as being within the same table." *Id.* at 52.

Microsoft's position is based upon the Packed Description in

SYS.TABLES allegedly defining the column type of any particular table, and that the columns are each represented as a row in the SYS.COLUMNS table. *See* Pet. 52–53, Ex. 1022 ¶ 336. We are persuaded that this evidence, even when read as Microsoft suggests, does not meet the single table requirement of base claim 31as discussed in section III.A.3. Accordingly, we conclude that Microsoft has not shown by a preponderance of the evidence that claim 36 would have been obvious over Chang and Goldberg.

F. Obviousness over Chang and Anderson, or over Chang, Anderson, and Horn

For the reasons given below, after consideration of Microsoft's Petition, Enfish's Preliminary Response, and PO Response, and the evidence cited therein, Microsoft has not shown, by a preponderance of the evidence, that claim 37 is unpatentable as obvious over the combination of Chang and Anderson, nor has Microsoft shown that claim 42 is unpatentable as obvious over the combination of Chang, Anderson, and Horn.

1. Anderson

Anderson discloses an electronic spreadsheet having numbered columns, for example 1, 2, 3 . . . etc., and letters referencing columns, for example A, B, C . . . etc. Ex. 1012, 1:51–65. The intersection of any row and column, for example B2, defines an addressable storage location, i.e. a "cell" for holding text and numeric information. *Id.* Anderson also teaches that spreadsheet cells can store formulas applying calculations to numbers stored in spreadsheet cells. *Id.* at 2:3–6. Anderson explains that "[i]n this fashion, cell references can serve as variables in an equation, thereby allowing precise mathematical relationships to be defined between cells." *Id.* at 6–9.

2. Claim 37

In support of this asserted ground of unpatentability, Microsoft relies on the Hosking Declaration (Ex. 1022 ¶¶ 342–353), explaining how each limitation is disclosed in Chang and Anderson. Pet. 53-54.

Microsoft asserts that Anderson teaches a spreadsheet program having notebooks, i.e., tables, with cells, rows and columns, where the cells have pointers to other columns. Pet. 53–54. Dr. Hosking testifies that one of skill in the art would understand spreadsheet programs as a simple form of a database similar to relational databases. Ex. 1022, ¶ 347, Further, Dr. Hosking states that one of skill in the art would therefore recognize it as obvious to combine the cell pointer functions in such spreadsheets with the relational tables in Chang, so that a cell in Chang's table would contain a plurality of pointers to other columns which contain defined values. *Id.* ¶ 347-351 (citing Ex. 1012, Figs. 4J, 4G).

Enfish argues that a cell address in a spreadsheet, for example the intersection of column A and row 2, "A2," shown in Anderson's Figure 4H, is not a "pointer" in a relational database, but "a cell reference [that] merely serves as a variable in the formula and the data stored within this referenced cell (i.e., cell A2) is used in the calculation." Prelim. Resp. 48–49. Enfish asserts specifically that "[t]he term "pointer" is known in the art to mean a variable that stores an address to a location where an object resides. The formula variables in Anderson are not pointers." PO Resp. 54. Enfish relies upon its Declarant, Dr. Jagadish, to explain that one of ordinary skill in the art of relational databases would not have understood a cell reference in Anderson's spreadsheet program, to be a "pointer," in a relational database as recited in the '775 patent. Ex. 2007 ¶ 215. Dr. Jagadish refers to three

textbooks which explain that [t]he plain and ordinary meaning of the term "pointer" is a variable that stores the address where another object resides." *Id.* ¶ 210. We credit Dr. Jagadish's testimony and evidence as to the meaning of the word "pointer" as it is understood by one of skill in the art because it is consistent with usage of the term "pointer" in the '775 patent. The '775 patent, although it does not give a specific definition, describes for example an OID used as a pointer to an object, where "the 'Employed By' column 126 is synchronized with the 'Employees' column by an OID pointer in the 'Synchronize With' column 144 to the 'Employees' column, represented by row 139." Ex. 1001, 10:8–12, *see also, Id.* at 12: 22–32.

Microsoft does not explain sufficiently or provide evidence that a cell reference for incorporating a variable stored in the cell into a formula in a spreadsheet would have been understood as a variable that stores an address to a location where an object resides in the database, at the time of the invention of the subject matter of the '775 patent.

We conclude that Microsoft has not shown by a preponderance of the evidence that claim 37 would have been obvious over Chang and Anderson.

3. Claim 42

For claim 42, Microsoft relies on the combination of Chang, Anderson, and Horn. Pet. 58. Microsoft contends that Horn discloses row-to-row pointers. Pet. 43. Claim 42, however, contains the same limitations as claim 37, therefore, Chang and Anderson do not meet the limitations of claim 42 for the same reasons discussed above in regard to claim 37. Accordingly, Microsoft has not shown by a preponderance of the evidence that claim 42 would have been obvious over Chang, Anderson, and Horn.

G. Obviousness over Chang and Covey

For the reasons given below, after consideration of Microsoft's Petition and Enfish's Preliminary Response, and the evidence cited therein, Microsoft has not shown, by a preponderance of the evidence, that claim 40 is unpatentable as obvious over the combination of Chang and Covey.

1. Covey

Covey is titled "Method for Creating and Maintaining a Database for a Dynamic Enterprise" and discloses a method of constructing a database which records a "token" to identify a database object, where a token is created for each event and includes a field corresponding to the event date. Ex. 1014. Abst. The database thus creates a record that preserves the historical state conditions of the database. *Id.* at 8:21–26.

2. Claim 40

Microsoft argues that dependent claim 40 is obvious in view of Chang and Covey. Microsoft concedes that Chang fails to disclose an OID having a session identification number and a timestamp. Pet. 55. Microsoft asserts that claim 40 is made obvious by Covey's disclosure of a relational database associating time values and session identifiers with database objects. *Id.* (citing Ex. 1014 at Abst., Figs. 12-13, 202). Microsoft argues that "[b]ecause Chang and Covey both address the same technical issues and disclose closely related subject matter, a person having ordinary skill in the art would be motivated to combine Chang and Covey." *Id.* at 55–56 (citing Ex. 1022 ¶¶ 36[6] –372. Dr. Hosking states that based on Chang's disclosure of variable length OID's and Patent Owner's contention that a text field, like a column name, can serve as an OID, it therefore "would be obvious, given this description, to add other types of identifying information

such as a session identifier and timestamp to each column." Ex. 1022 ¶ 370.

Enfish counters that Microsoft's statement that the references are analogous art is not sufficient to support obviousness. Prelim. Resp. 52. Microsoft does not, in the Petition, provide any reason for the combination of Chang and Covey apart from asserting the references are analogous art and address similar technical issues. Pet. 55–56. This is not a sufficient reason with rational evidentiary underpinnings to support the combination of Chang and Covey.

Microsoft cites to Dr. Hosking's testimony, apparently to provide the necessary reasoning and rational underpinnings. *Id.* at 56 (citing Ex. 1022 ¶ 36[6]–372). We are not persuaded by this testimony. Dr. Hosking's explanation of Chang's OID being variable length, and capable of being a text field, does not explain why one or ordinary skill would have looked to Covey to include a session identification and time stamp in an OID. At best, this reasoning explains that Chang's OID could accommodate the text and length of an OID including a session identification and time stamp. Dr. Hosking's statement, "I believe that it would be obvious, given [Chang's] description, to add other types of identifying information such as a session identifier and timestamp to each column," is hindsight. *See* Ex. 1022 ¶ 370. It is not, a sufficient reason with rational evidentiary underpinnings to support the combination of Chang and Covey.

We conclude that Microsoft has not shown by a preponderance of the evidence that claim 40 would have been obvious over Chang and Covey.

H. Secondary Considerations

The factual inquiries for obviousness include secondary considerations based on evaluation and crediting of objective evidence.

Graham v. John Deere Co., 383 U.S. 1, 17 (1966). However, to accord substantial weight to objective evidence requires the finding of a nexus between the evidence and the merits of the claimed invention. *In re GPAC Inc.*, 57 F.3d 1573, 1580 (Fed. Cir. 1995); *see also In re Huang*, 100 F.3d 135, 140 (Fed. Cir. 1996) ("success is relevant in the obviousness context only if there is proof that the sales were a direct result of the unique characteristics of the claimed invention.").

Enfish contends that the claimed invention received industry accolades, including praise from Microsoft, satisfied a long-felt need, resulted in success where others had failed, as well as commercial success and copying. PO Resp. 61–65. Enfish points to features of claim 31 that it contends resulted in the objective indicia of success to which Enfish refers. *Id.* at 61. We are not convinced by this argument. Claim 31 of the '775 patent is challenged as anticipated by Chang. Secondary considerations do not weigh into determinations regarding anticipation. *Cohesive Techs., Inc. v. Waters Corp.*, 543 F.3d 1351, 1364 (Fed. Cir. 2008).

In addition, we are not persuaded by secondary considerations here because Enfish has not shown a nexus between any of the accolades or successes it says occurred and the use of the pointer or pointer searching function.

Enfish asserts that Microsoft failed at developing a suitable search engine. PO Resp. 65–66. We are not persuaded by this argument. The statements of Bill Gates and others at Microsoft submitted relied on by Enfish in support of this assertion are not tied sufficiently to any claim at issue in this proceeding. Instead, the statements broadly refer to search engines. None of the statements reference two-way pointers, folder objects

or searching for pointers or OID's.

Enfish further submits evidence to show commercial success. PO Resp. 66. The evidence Enfish cites refers to how many users downloaded Enfish's software and to the planning of a collaborative effort as set forth in one paragraph of its business plan. *See* Ex. 2025, 2. Enfish's evidence, however, does not indicate that these users paid for or actually used the downloaded software. The evidence also does not indicate how the number of downloads indicates commercial acceptance, for example, as compared to downloads of other software at the time. Additionally, a planned collaborative effort does not indicate the results of the collaboration. Enfish's evidence does not establish commercial acceptance or financial success. *See In re Fielder*, 471 F.2d 640, 644 (CCPA 1973). Thus, we are not persuaded by Enfish's evidence of commercial success.

I. Motion to Correct Patent Owner Response

After institution of trial, Enfish timely filed a Patent Owner Response (Paper 24), along with the Jagadish Declaration (Ex. 2007). On September 16, 2014, Enfish filed an unopposed motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration, which Enfish contends correct only typographical errors and erroneous citations. Paper 40. We grant Enfish's September 16, 2014 motion.

J. Joint Stipulation

On November 14, 2014, the parties filed a joint stipulation requesting that we expunge confidential versions of exhibits 2049–2058 and 2060–2065. Paper 57. The parties contend that Microsoft withdraws its motion to seal (Paper 28) provided that we expunge the confidential versions. Paper 57. Microsoft agrees that the sealed version of Exhibit 2059 may be

unsealed. *Id.* We hereby grant the motion and expunge only confidential versions of exhibits 2049–2058 and 2060–2065.

K. Motion to Exclude

On November 3, 2014, Microsoft filed a motion to exclude Exhibit 2071, the Declaration of Dr. Sharad Mehrotra ("Mehrotra Declaration") and two paragraphs of the Declaration of Louise Wannier ("Wannier Declaration," Exhibit 2077 ¶¶ 32, 33). Paper 48.

Regarding the Mehrotra Declaration, we agree with Microsoft's assertion that Dr. Mehrotra provides only conclusory opinions and, therefore, we do not rely on it in this Decision. 37 C.F.R. § 42.65(a). Because Microsoft has not argued persuasively any other reason to exclude the Mehrotra Declaration, we deny Microsoft's request to exclude it.

Regarding the Wannier Declaration, we disagree with Microsoft that "Patent Owner has no basis to file the Wannier Declaration as supplemental evidence because Microsoft has not moved to exclude the Armon Declaration." Paper 48, 4–5. Patent Owner is entitled to submit supplemental evidence in response to Microsoft's objection. 37 C.F.R. § 42.64(b)(2). Microsoft further contends that the Wannier Declaration inserts untimely, conclusory, and improper technical opinions. Paper 48, 5. Patent Owner contends that paragraphs 32 and 33 do not exceed the scope because they are submitted to support admissibility. Paper 60, 3–4. We agree with Microsoft that the Wannier Declaration provides conclusory technical opinions, and, therefore, we do not rely upon it. Because Microsoft has not argued persuasively any other reason to exclude paragraphs 32 and 33 of the Wannier Declaration, we deny Microsoft's request to exclude it.

IV. CONCLUSION

We conclude that Microsoft has demonstrated by a preponderance of the evidence that (1) claims 31 and 41 of the '775 patent are anticipated by Chang, and (2) claim 45 of the '775 patent is obvious over the combination of Chang and Webb.

We further conclude that Microsoft has not shown that claims 32–40 and 42–44 of the '775 patent are unpatentable as obvious. In addition, we terminate this proceeding with respect to claims 1–15 under 37 C.F.R. § 42.72.

V. ORDER

For the reasons given, it is

ORDERED that claims 31, 41, and 45 of U.S. Patent No. 6,163,775 are determined by a preponderance of the evidence to be unpatentable;

FURTHER ORDERED that this proceeding is TERMINATED, under 37 C.F.R. § 42.72, with respect to claims 1–15;

FURTHER ORDERED Enfish's motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration (Paper 38) is GRANTED;

FURTHER ORDERED that Microsoft's motion to exclude (Paper 48) is DISMISSED;

FURTHER ORDERED that confidential versions of Exhibits 2049–2058 and 2060–2065 are EXPUNGED;

FURTHER ORDERED Microsoft's motion to seal is DISMISSED; and

FURTHER ORDERED that because this is a final written decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

For PETITIONER:

Amy E. Simpson
Chad Campbell
PERKINS COIE LLP
ASimpson@perkinscoie.com
CCampbell@perkinscoie.com

For PATENT OWNER:

Frank Pietrantonio
Orion Armon
COOLEY LLP
fpietrantonio@cooley.com
oarmon@cooley.com
zpatdcdocketing@cooley.com

PTAB Decision in IPR2013-00560

Trials@uspto.gov
571.272.7822

Paper 63
Entered: March 2, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

ENFISH, LLC,
Patent Owner.

Case IPR2013-00560
Patent 6,163,775

Before THOMAS L. GIANNETTI, BRYAN F. MOORE,
and BARBARA A. PARVIS, *Administrative Patent Judges.*

PARVIS, *Administrative Patent Judge.*

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.73

I. INTRODUCTION

A. Background

Microsoft Corporation ("Microsoft") filed a petition to institute an *inter partes* review of claims 17–23 and 47–53 of U.S. Patent No. 6,163,775 ("the '775 patent"). Paper 1 ("Pet."). We instituted trial for claims 17–23 and 47–53 of the '775 patent on certain grounds of unpatentability alleged in the Petition. Paper 14 ("Decision to Institute" or "Inst. Dec.").

After institution of trial, Patent Owner, Enfish, LLC ("Enfish"), filed a Patent Owner Response, along with a Declaration by Dr. H.V. Jagadish ("Jagadish Declaration").

On September 16, 2014, Enfish filed an unopposed motion to correct typographical errors in both papers and filed therewith the corrected Patent Owner Response ("PO Resp.," Paper 40) and the corrected Jagadish Declaration (Ex. 2007). We grant Enfish's motion to correct both papers.[1] Microsoft filed a Reply ("Pet. Reply"). Paper 35.

A consolidated hearing for IPR2013-00559, IPR2013-00560, IPR2013-00561, IPR2013-00562, and IPR2013-00563 was held on December 3, 2014. The transcript of the consolidated hearing has been entered into the record. Paper 62 ("Tr.").

We have jurisdiction under 35 U.S.C. § 6(c). This final written decision is issued pursuant to 35 U.S.C. § 318(a).

Microsoft has shown by a preponderance of the evidence that claims 47–50 of the '775 patent are unpatentable. Microsoft has not shown that

[1] For ease of reference, throughout we refer to Enfish's corrected Patent Owner Response (Paper 40) and corrected Jagadish Declaration (Exhibit 2007), both filed on September 16, 2014.

claims 51–53 of the '775 patent are unpatentable. For the reasons discussed below, we are unable to reach a determination on the alleged grounds of unpatentability over prior art for claims 17–23. Accordingly, we terminate this proceeding with respect to claims 17–23 under 37 C.F.R. § 42.72.

B. Additional Proceedings

In addition to this petition, Microsoft has filed a petition challenging the patentability of claims 1–16, 24–46, and 54–60 of the '775 patent. *See Microsoft Corp. v. Enfish, LLC*, IPR2013-00559; IPR2013-00561. Microsoft indicates that claims of the '775 patent have been asserted against Microsoft in *Enfish LLC v. Microsoft Corporation, et al.*, Case No. 12-cv-7360 MRP, in the Central District of California ("California case"). Pet. 2. Microsoft further contends that a final judgment against Enfish has been entered in the California case. Ex. 1168.

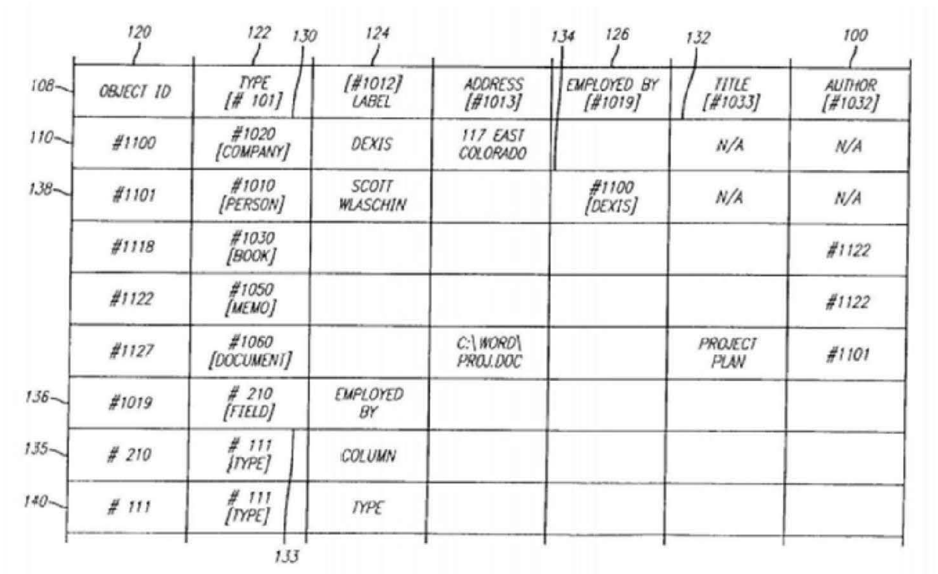
Microsoft also has filed two petitions challenging the patentability of claims 1–60 of U.S. Patent 6,151,604 ("the '604 patent").[2] *See Microsoft Corp. v. Enfish, LLC*, IPR2013-00562; IPR2013-00563.

C. The '775 Patent

The '775 patent (Ex. 1101) is titled, "Method and Apparatus Configured According to a Logical Table Having Cell and Attributes Containing Address Segments," and generally relates to a system and method for data storage, manipulation, and retrieval in a logical table of a database. This table is a logical structure, not a physical structure, stored in the memory. Ex. 1101, 6:39–41.

[2] The '604 patent and the '775 patent both issued from continuations of Application No. 08/383,752, filed March 28, 1995, now U.S. Patent No. 5,729,730. Pet. 5.

Figure 3 is reproduced below:



OBJECT ID	TYPE [# 101]	[#1012] LABEL	ADDRESS [#1013]	EMPLOYED BY [#1019]	TITLE [#1033]	AUTHOR [#1032]
#1100	#1020 [COMPANY]	DEXIS	117 EAST COLORADO		N/A	N/A
#1101	#1010 [PERSON]	SCOTT WLASCHIN		#1100 [DEXIS]	N/A	N/A
#1118	#1030 [BOOK]					#1122
#1122	#1050 [MEMO]					#1122
#1127	#1060 [DOCUMENT]		C:\WORD\PROJ.DOC		PROJECT PLAN	#1101
#1019	# 210 [FIELD]	EMPLOYED BY				
# 210	# 111 [TYPE]	COLUMN				
# 111	# 111 [TYPE]	TYPE				

Figure 3 of the ’775 patent illustrates a structure of a logical table. Ex. 1101, 3:36–37. As depicted by Figure 3 of the ’775 patent, above, table 100 is defined by rows 108, 110, 138, 136, 135, and 140, and columns 120, 122, 124, and 126. *Id.* at Fig. 3. The intersection of a row and a column defines a cell in the table. *Id.* at 6:48–49. Each column corresponds to an attribute spanning various records. *Id.* at 6:46. An attribute is a single class description, such as an employer, denoted in column 126 of Figure 3, for example, by the text “Employed By.” *Id.* at 7:25–26.

Each row corresponds to a record spanning various attributes. Ex. 1101, 6:45–46. For example, row 110 corresponds to a company as shown in cell 130. *Id.* at 6:62–64.

D. Illustrative claims

Of the challenged claims, the independent claims are 17 and 47. Each of claims 18–23 depend, directly or indirectly, from claim 17. Claims 48–53

depend, directly or indirectly, from claim 47. Claim 47 illustrates the claimed subject matter and is reproduced below:

> 47. A method for storing and retrieving data in a computer system having a memory, a central processing unit, and a display, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of cells, each said cell having a first address segment and a second address segment;
>
> a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;
>
> a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record; and
>
> indexing data stored in said table.

E. The Prior Art References Supporting Alleged Unpatentability

Microsoft relies on the following references:

MICROSOFT® VISUAL BASIC™ PROGRAMMING SYSTEM FOR WINDOWS™ VERSION 3.0 (1993) ("Visual Basic," Ex. 1104).

Jensen et al., U.S. Patent No. 5,615,362 (issued Mar. 25, 1997) ("Jensen," Ex. 1106).

GERARD SALTON & MICHAEL J. MCGILL, INTRODUCTION TO MODERN INFORMATION RETRIEVAL (1983) ("Salton," Ex. 1107).

Smith et al., U.S. Patent No. 5,181,162 (issued Jan. 19, 1993) ("Smith '162," Ex. 1108).

F. The Pending Grounds of Unpatentability

The following chart summarizes Microsoft's patentablilty challenges:

Reference[s]	Basis	Claims Challenged
Visual Basic	§ 102(b)	17, 18, 47, and 48
Visual Basic and Jensen	§ 103	19, 20, 49, and 50
Visual Basic and Salton	§ 103	23 and 53
Visual Basic, Jensen, and Salton	§ 103	21 and 51
Visual Basic and Smith '162	§ 103	22 and 52

In support of the above-referenced grounds of unpatentability, Microsoft relies on the Declaration of Dr. Hosking (Ex. 1119, "Hosking Declaration").

II. CLAIM CONSTRUCTION

A. Legal Standard

In an *inter partes* review, claim terms in an unexpired patent are interpreted according to their broadest reasonable construction in light of the specification of the patent in which they appear. 37 C.F.R. § 42.100(b); *see also In re Cuozzo Speed Techs., LLC.*, No. 14-01301, slip op. at 16, 19 (Fed. Cir. Feb. 4, 2015) ("Congress implicitly adopted the broadest reasonable interpretation standard in enacting the AIA," and "the standard was properly adopted by PTO regulation."). Under the broadest reasonable construction standard, claims are to be given their broadest reasonable interpretation consistent with the specification, and the claim language should be read in light of the specification as it would be interpreted by one of ordinary skill in

the art. *In re Am. Acad. of Sci. Tech. Ctr.*, 367 F.3d 1359, 1364 (Fed. Cir. 2004). Also, we must be careful not to read a particular embodiment appearing in the written description into the claim, if the claim language is broader than the embodiment. *See In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993) ("[L]imitations are not to be read into the claims from the specification.").

B. Overview of the Parties' Positions

Microsoft contends that this case involves a computer-implemented invention. Pet. 6. Microsoft also contends that the challenged independent claims invoke means-plus-function claiming, but the '775 patent fails to disclose an algorithm, which is an issue under § 112, sixth paragraph, that cannot be addressed in this proceeding. Pet. 11. In the Decision to Institute, we invited Enfish to direct us to the specific portions of the specification that clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. Inst. Dec. 10. For the reasons discussed below, we determine that Enfish in its response does not identify sufficient corresponding structure, as required under 35 U.S.C. § 112, sixth paragraph, for the "means for configuring said memory according to a logical table," recited in claim 17.

Regarding terms in the method claims, in the Decision to Institute, we provided constructions for "logical table," "object identification number," and "attribute set" which are shown in the table below. Inst. Dec. 10–13. We also determined that no express construction is needed for the following terms: "memory," "a cell having a first address segment and a second address segment," and "records." Inst. Dec. 14.

Claim Term or Phrase	Construction in the Decision to Institute
"logical table"	"[W]e construe the term "table" to mean: 'a structure of a database comprising rows and columns.'" Inst. Dec. 11. "We determine no express construction of 'logical' is needed for this decision." *Id.*
"object identification number"	"[W]e construe 'object identification number' in light of the specification to mean: 'an array of bits that define an object.'" *Id.* at 12.
"attribute set"	"[A] name or characteristic corresponding to a column." *Id.*

Enfish contends that our construction for "object identification number" is incomplete. PO Resp. 15. We evaluate Enfish's contention below. Enfish also provides a proposed construction for "anchor," recited in claims 21 and 51. PO Resp. 22. Because we determine that Microsoft has not met its burden with respect to these claims for other reasons, no construction of this term for the purposes of this Decision is necessary.

We discern no reason, based on the complete record now before us, to change our constructions in the Decision to Institute of "logical table," "object identification number," and "attribute set."

C. Analysis of the Parties' Claim Construction Positions

1. Means for configuring said memory according to a logical table

Independent claim 17 includes the limitation "means for configuring said memory according to a logical table." In the Decision to Institute, we agreed with Microsoft that under the broadest reasonable interpretation, the function for the means for configuring is "configuring memory according to a logical table." Inst. Dec. 10. Additionally, we considered the corresponding structure for the recited function as including a general

purpose computer. *Id.* Enfish does not challenge persuasively either of these determinations; however, Enfish identifies portions of the specification that Enfish contends provide algorithmic support for the recited function. PO Resp. 23–25. In particular, Enfish contends that the '775 patent discloses a four-step algorithm that is linked to the recited function of configuring memory according to a logical table. *Id.*

"[T]he corresponding structure for a § 112 ¶ 6 claim for a computer-implemented function is the algorithm disclosed in the specification." *Aristocrat Techs. Austl. Party Ltd. vs. Int'l Game Tech.*, 521 F.3d 1328, 1333 (Fed. Cir. 2008) (quoting *Harris Corp. v. Ericsson Inc.*, 417 F.3d 1241, 1249 (Fed. Cir. 2005)). Additionally, specific portions of the specification must clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. *See Medical Inst. & Diag. Corp. v. Elektra AB*, 344 F.3d 1205, 1211 (Fed. Cir. 2003). For the reasons set forth below, we conclude that the four steps and other '775 patent specification portions identified by Enfish do not describe an algorithm for the recited function of "configuring memory according to a logical table." We further conclude that Enfish has not clearly linked or associated the recited function to the four steps or the portions of the '775 patent specification that Enfish identifies.

For algorithmic support, Enfish identifies disparate excerpts of the '775 patent specification, which do not link or associate clearly a computer program or algorithm to the function corresponding to the claimed means for configuring said memory according to a logical table. For example, the first step, "[c]reate, in a computer memory, a logical table" (PO Resp. 23–24) appears to be similar to the recited function of configuring memory

according to a logical table, but the first step is not found in the '775 patent specification. Additionally, none of the portions of the '775 patent specification that Enfish cites for this first step provide an algorithm or computer program for performing the recited function of "configuring memory according to a logical table" or clearly link or associate any algorithm or program to this recited function. PO Resp. 24 (citing Ex. 1101, Abstract, 2:59–63, 6:38–46, Figs. 1, 3, 9). These portions describe that an already-formed table has rows and columns, without describing how memory is configured to create a logical table having rows and columns. Ex. 1101, 2:59–63. One portion states that memories "need not store" the table contiguously, but fails to describe an algorithm or computer program for configuring memory such that a logical table is not stored contiguously. *Id.* at 6:38–46.

Enfish's three remaining steps and the other '775 patent specification portions identified by Enfish fail to remedy these deficiencies. The second step is: "[a]ssign each row and column an object identification number (OID) that, when stored as data, can act as a pointer to the associated row or column." PO Resp. 24. One of the portions of the '775 patent specification cited by Enfish for the second step indicates "the system must generate a unique OID when columns and rows are formed." Ex. 1101, 8:15–16. The remainder of the identified portions relate to assigning an OID or indicate that an OID "may be used" as a pointer, without describing an algorithm for forming columns and rows of a table or showing how assigning an OID relates to steps for forming columns and rows. Ex. 1101, Abstract, 2:60–61, 6:50–57, 7:1–2, 8:18–60, Figs. 3, 4. Additionally, the identified portions discuss assigning a numeric value to an OID in the form of a bit array, but

fail to describe how to configure memory such that an OID may be used as a pointer. *Id.*

The third step is a feature of claim 1 (PO Resp. 11). That claim is not challenged in this proceeding. Furthermore, the portions of the '775 patent specification cited by Patent Owner for the third step (PO Resp. 25 (citing Ex. 1101, Abstract, 2:66–3:2, 6:47–48, 7:25–31, Fig. 3)) indicate that a table has a row that corresponds to columns, such as a header row, which is a generic feature of an already-formed table. These '775 patent specification portions identified by Enfish (*id.*), however, do not describe how to form a table with this feature or link or associate this feature to the recited function.

The fourth step, i.e., storing and accessing data in cells (PO Resp. 25), is performed in an already-configured table and, therefore, occurs after the recited function of configuring a table has occurred. Nonetheless, the '775 patent specification portions identified by Enfish (*id.* (citing Ex. 1101, Abstract, 2:63–66, 6:48–49, 6:61–62, 7:9–10, 7:23–24, 11:52–62, Fig. 10)) suffer from the same deficiencies noted above.

We conclude that the four steps and '775 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table. The portions of the '775 patent specification identified by Enfish describe an already-formed table having generic characteristics, such as columns, rows, identifiers, and a header row. *See, e.g.*, Ex. 1101, Fig. 3. The description, however, does not disclose any algorithm or computer program for forming this table. Additionally, the description identified by Enfish is not linked or associated clearly with the recited function of "configuring memory according to a logical table."

Enfish also relies on Dr. Jagadish's Declaration to show that the '775

patent specification provides an algorithm and clearly links the algorithm to the recited function. PO Resp. 23–25 (citing Ex. 2007 ¶¶ 68–76). The Jagadish Declaration, however, relies on similar portions of the '775 patent specification to those cited in Patent Owner's Response. For the reasons given, the Jagadish Declaration does not support Enfish's assertion. Dr. Jagadish further asserts, "[i]t is also my opinion that one of ordinary skill in the art would understand how to implement those algorithm steps using techniques and resources that were available at the time the '775 Patent was filed." Ex. 2007 ¶ 69. Enfish, however, cannot rely on the knowledge of one skilled in the art to address the deficiencies noted above. *See Function Media, LLC v. Google Inc.*, 708 F.3d 1310, 1319 (Fed. Cir. 2013) ("Having failed to provide any disclosure of the structure . . . FM cannot rely on the knowledge of one skilled in the art to fill in the gaps.")

Because we conclude that the four steps and '775 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table, we terminate this proceeding with respect to the claims that recite this means-plus-function limitation. As explained in *BlackBerry Corporation v. Mobile Media Ideas, LLC*, Case IPR2013-00036 (PTAB Mar. 7, 2014) (Paper 65), the specification must disclose enough of a specific algorithm to provide the necessary structure under § 112, sixth paragraph. In the circumstance when the specification of the challenged patent lacks sufficient disclosure of structure under 35 U.S.C. § 112, sixth paragraph, the scope of the claims cannot be determined without speculation and, consequently, the differences between the claimed invention and the prior art cannot be ascertained. *Id.* For the reasons given, we determine that independent claim 17 is not amenable to construction and,

thus, we terminate this proceeding with respect to claim 17 under 37 C.F.R. § 42.72. Because claims 18–23 depend, directly or indirectly, from claim 17, we also terminate this proceeding with respect to these claims under 37 C.F.R. § 42.72.

2. Other Means-Plus-Function Terms

Because we determine that this proceeding should be terminated as to claims 18–23 for the reasons discussed above, we need not construe other means-plus-function terms appearing in those claims for the purposes of this Decision.

3. Object Identification Number (OID)

Independent claim 47 recites "object identification number (OID)" and "OID." In the Decision to Institute, we construed "object identification number" in light of the specification to mean: "an array of bits that define an object." Inst. Dec. 12 (citing Ex. 1101, 3:38–40, 8:44–46).

Enfish "agrees in principle" with our construction, but contends that the construction is incomplete. PO Resp. 15. Enfish contends that "OID" should be defined further as "a unique, immutable, and system-generated value that identifies an object." *Id.* Microsoft contends that Enfish seeks to read into the claims extraneous limitations that are unsupported by either intrinsic or extrinsic evidence. Pet. Reply 2. Microsoft additionally contends that Enfish's proposed construction does not provide an appropriate context for the additional limitations. *See, e.g.*, Pet. Reply 3.

At the heart of Enfish's contention is an assertion that an OID will not function properly if it is not unique, immutable, and system generated. PO Resp. 16–21. For example, Enfish, relying on Dr. Jagadish, contends that an

OID must be a unique value because "if an OID were not unique the database would be non-functional because objects could not be reliably retrieved." PO Resp. 16 (citing Ex. 2007 ¶ 51). Dr. Jagadish cites to a portion of the specification stating that the OID is used for "exact retrieval" and states that without uniqueness, retrieval using an OID will result in more than one object. Ex. 2007 ¶ 51 (citing Ex. 1101, 1:60–64).

We determine that Enfish's proposed construction of OID introduces requirements beyond what is needed for an OID to identify data and retrieve data from memory. Enfish, for example, contends that "an OID is unique among all rows and columns of the logical table." PO Resp. 29. Additionally, claim 47 recites that an OID is included "to identify" each attribute set and each record, but does not specifically recite using an OID to retrieve data from memory.

Microsoft advances additional contentions regarding why Enfish's proposed construction should not be adopted. For example, Microsoft contends that Enfish's "unique" OID requirement would conflict with the patent, which describes a row in a table with an OID that has the same value as the OID of a column in the same table. Pet. Reply 3 (citing Ex. 1101, Fig. 3). Microsoft also contends that "immutable" should not be imported into the construction of OID because the term is absent from the specification, claims, and file history, and the specification is devoid of disclosure relating to immutability of an OID. Pet. Reply 3–4. As further support, Dr. Hosking states that an OID may change in certain circumstances and still retrieve information reliably. Ex. 1143 ¶¶ 21, 22.

We determine that adopting Enfish's proposed construction without further context would create ambiguity. Additionally, limitations are not to

be read into the claims from embodiments in the specification. *See In re Van Geuns*, 988 F.2d at 1184. Furthermore, we agree with Microsoft that Enfish's reliance on extrinsic evidence is misplaced. Enfish, for example, relies on excerpts of "text books" (PO Resp. 20) for its contention that an OID is "immutable," without showing sufficiently that the definition ascertained from these excerpts of extrinsic evidence is consistent with the definition that would be ascertained by reading the patent documents.

We, therefore, determine that the construction of "OID" in the Decision to Institute should not be changed.

III. ANALYSIS

A. Alleged Anticipation of Claim 47 by Visual Basic

For the reasons given below, after consideration of the arguments in Enfish's Patent Owner Response, and the evidence cited therein, we conclude that Microsoft has shown, by a preponderance of the evidence, that claim 47 is unpatentable as anticipated by Visual Basic.

1. Visual Basic

Visual Basic describes a programming system for the Windows™ operating system that enables programmers to create databases. Ex. 1104, PF 1, PG 453.[3] The programming system uses objects to represent tables of a database. *Id.* at PF 47. As described in Visual Basic, a table object is a logical representation of a physical table with records (rows) and fields (columns). *Id.* at LR 558.

---

[3] Citations in this decision to Exhibit 1104 refer to the volumes Programmer's Guide (PG), Language Reference (LR), and Professional Features (PF), respectively, as well as the page number within each volume.

Visual Basic also describes adding an index to a database, "[a]dding an index to your database can increase the speed with which you get access to information." Ex. 1104, PF 37. Visual Basic provides an example of an index with key fields Name, Last Name, City, and Zip Code. *Id.* at 38.

2. Claim 47

We have reviewed Microsoft's anticipation argument, supporting evidence, and the detailed claim analysis, which reads persuasively all elements of claim 47 onto the disclosure of Visual Basic. Pet. 33–46 (citing Ex. 1104, LR 162, 185, 280, 425, 484, 532, 558, PF 1, 3, 18, 37–39, 40, 47, 76, 80, PG 453, 471, 481, 482 ; Ex. 1119 ¶¶ 67–98). For example, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses configuring memory according to a logical table, as recited in claim 47. In the cited portions above, Visual Basic discloses storing data in a computer formatted as a table object, which is "a logical representation of a physical table." Ex. 1104, LR 558. The table object comprises "records (rows) and fields (columns)." *Id.*

As an additional example, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses each record and each attribute set including an OID, as recited in claim 47. In the cited portions above, Visual Basic describes assigning a primary key and that "[a] table's primary key is the determining factor when testing to see if the record is unique within the table." Ex. 1104, PF 39. According to Visual Basic "[Visual Basic] creates an index on the primary key of the table and uses it to find records and to create joins between tables." *Id.* Regarding each column including an OID, according to Visual Basic, "[e]ach Field in the Fields collection of a TableDef has a unique value in the OrdinalPosition

property." *Id.* at LR 425.

Furthermore, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses indexing data stored in the table, as recited in claim 47. In the cited portions above, Visual Basic discloses an index object used for creating indexes to "increase the speed" of finding records. Ex. 1104, PF 37, 38, PG 482.

Enfish contends that Microsoft has not met its burden of showing that Visual Basic discloses OIDs. PO Resp. 29–33. Enfish's contentions are based on its proposed construction of OID, which we decline to adopt for the reasons given above.

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 47 of the '775 patent is anticipated by Visual Basic.

3. Claim 48

Claim 48 depends directly from independent claim 47. Microsoft has shown by a preponderance of the evidence that Visual Basic describes the additional elements recited in claim 48 of searching the table for a key word and inserting into the table a record corresponding to the key word. Pet. 46–47 (citing Ex. 1104, PF 1, 37, PG 487, Fig. 20.6; Ex. 1119 ¶¶ 99–101). Accordingly, Microsoft has demonstrated by a preponderance of the evidence that claim 48 of the '775 patent is unpatentable, under 35 U.S.C. § 102(b), as anticipated by Visual Basic.

B. Obviousness of Claims 49 and 50 in view of Visual Basic and Jensen

For the reasons given below, after consideration of the Petition, Enfish's Patent Owner Response, and the evidence cited therein, Microsoft

has shown, by a preponderance of the evidence, that each of claims 49 and 50 is unpatentable as obvious over the combination of Visual Basic and Jensen.

1. Jensen

Jensen describes managing information retrieved from a structured database, such as a relational database, by constructing object instances in which each object instance has its own unique object identifier that provides a mapping between the object instance and at least one row in the structured database. Ex. 1106, Abstract. Jensen further describes a pointer that is an object instance attribute containing an address, such as a physical address, of another object instance. *Id.* at 7:33–35. Jensen also describes bidirectional pointers that are set up between a department instance and related employee instances. *Id.* at 10:9–11.

2. Claims 49 and 50

Claims 49 and 50 depend, directly or indirectly, from claim 48. Microsoft has shown by a preponderance of the evidence that Visual Basic describes the additional elements recited in claims 49 and 50. Pet. 47–52 (citing Ex. 1104, PF 69, PG 487, 489, Fig. 20.6; Ex. 1106, 7:29–35, 8:24–29, 8:50–10:17, 10:45–48, 10:62–11:26, Fig. 2, 3; Ex. 1119 ¶¶ 385–396, 397–404).

Enfish contends that Microsoft's motivation-to-combine arguments are insufficient. PO Resp. 35 (citing Ex. 2007 ¶ 231). Dr. Jagadish states "[d]ue to the dissimilarity of environments between [Visual Basic] and Jensen (a relational database employing unidirectional pointers in an indexing structure vs. linking related objects in an object cache), extensive redesign of the system in [Visual Basic] would be required." Ex.

2007 ¶ 231. In contrast to Dr. Jagadish's statement, however, Jensen teaches an object-oriented application for managing information retrieved from a structured database, such as a relational database. Ex. 1106, Abstract.

Enfish further contends that Microsoft does not explain why one of ordinary skill in the art would have been motivated to apply the pointers of Jensen to the indexing structure in Visual Basic. PO Resp. 35. We, however, are persuaded that Microsoft has shown a sufficiently articulated reason with rational underpinning to support obviousness (Pet. 48–49 (citing Ex. 1119 ¶¶ 387–404)). Microsoft, for example, states "[i]t was *well known* in the art at the time that indexes such as the index disclosed in [Visual Basic], maintain object identification numbers of the cell for locating data." Pet. 49 (emphasis added). Microsoft continues, "Jensen discloses containing the address in a pointer, *e.g.*, a physical memory address (*i.e.*, OIDs)." *Id.* (citing Ex. 1106, 7:29–35). Dr. Hosking testifies "[i]t was *well known* in the art at the time that Index Objects locate data by maintaining object identification numbers in the internal pointers as disclosed in Jensen." Ex. 1119 ¶ 403 (emphasis added). Dr. Hosking further testifies "[a] person of ordinary skill in the art would be motivated to combine Jensen's teaching of maintaining the OIDs in a pointer with [Visual Basic's] disclosure of identifying the logical columns / rows with the OIDs." Ex. 1119 ¶ 404. The predictable use of familiar prior art elements according to their established functions renders the recited invention obvious. *See KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 417 (2007).

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claims 49 and 50 of the '775 patent are unpatentable as obvious over the combination of Visual Basic and Jensen.

C. Obviousness of Claim 53 in view of Visual Basic and Salton and Obviousness of Claim 51 in view of Visual Basic, Jensen, and Salton

For the reasons given below, Microsoft has not shown that claims 51 and 53 are unpatentable as obvious.

1. Salton

Salton describes information retrieval. Ex. 1107, 7. In particular, Salton describes speeding up a search for information by developing an index. *Id.* at 16. An index of Salton includes values for each key for records in a file. *Id.* at 17.

2. Claim 53

Enfish contends that Microsoft's motivation-to-combine contentions are insufficient. PO Resp. 40–42. Dr. Jagadish states, "it is my opinion that a POSITA would not have been motivated to import, into [Visual Basic], the teachings in Salton regarding indexing architectures." Ex. 2007 ¶ 242. Dr. Jagadish provides various reasons including a lack of teachings in Visual Basic, difficulties described in Salton, as well as difficulties that would have been known to one of ordinary skill in the art. Ex. 2007 ¶¶ 240–242.

Microsoft states that one of ordinary skill in the art would have been motivated to combine Salton with Visual Basic to employ the indexing and text analysis techniques taught by Salton in the object database systems of Visual Basic. Pet. 50 (citing Ex. 1119 ¶¶ 420–422). We are not persuaded that this is a sufficient rationale for combining Visual Basic and Salton, because Microsoft does not provide adequate evidentiary support. Paragraphs 420 through 422 of Dr. Hosking's Declaration simply reiterate claim language. Although Dr. Hosking additionally testifies that Salton

describes speeding up a search through indexing techniques and identifies particular techniques in Salton to be combined with Visual Basic (Ex. 1119 ¶¶ 438–444), Dr. Hosking does not indicate that speeding up searching using indexing is a reason to combine Visual Basic and Salton. Dr. Hosking, instead, states that Visual Basic and Salton are in the same field and provides a conclusion that one of ordinary skill in the art would have been motivated to combine Visual Basic and Salton. Ex. 1119 ¶ 444. Furthermore, in its contentions regarding another challenged claim, Microsoft relies on Visual Basic for describing indexing. *See* Pet. 45 (Ex. 1104, PF 37, 38). In light of these other contentions, Microsoft does not explain sufficiently why one of ordinary skill in the art would have been motivated to employ Salton's indexing techniques with Visual Basic.

As discussed above, Dr. Jagadish disagrees with Dr. Hosking. Ex. 2007 ¶¶ 240–242. In its Reply, Microsoft does not address Enfish's contentions or the testimony provided by Dr. Jagadish . Pet. Reply 11–12.

3. Claim 51

With respect to claim 51, Microsoft, relying on Dr. Hosking, further states that "[o]ne of ordinary skill in the art would have been motivated to combine Salton with [Visual Basic] and Jensen to employ the indexing and text analysis techniques taught by Salton in the object database systems of [Visual Basic] and Jensen to improve data searching and retrieval efficiency." Pet. 52 (citing Ex. 1119 ¶¶ 407–410). Dr. Hosking, however, states that the references relate to data managing and then states his conclusion that one of ordinary skill in the art would have been motivated to combine them. Ex. 1119 ¶ 410.

4. Conclusion

We determine that Dr. Hosking's statements are not a sufficient rationale for combining these references because he bases his conclusions on simple statements that the references are in the same field. Microsoft does not provide sufficient additional reasons, such as expert testimony that the combinations of prior art references are of familiar elements according to known methods that yield no more than predictable results. We, therefore, conclude that Microsoft has not demonstrated that the teachings would have been combined by a person of ordinary skill.

For the foregoing reasons, we determine that Microsoft has not set forth a sufficient articulated reason with a rational underpinning to support a showing of obviousness by a preponderance of the evidence. We, therefore, conclude that Microsoft has not shown that claims 51 and 53 of the '775 patent are unpatentable as obvious.

D. Obviousness of Claim 52 in view of Visual Basic and Smith '162

For the reasons given below, Microsoft has not shown that claim 52 is obvious.

1. Smith '162

Smith '162 describes an object-oriented document management and production system. Ex. 1108, Abstract. Stored objects are organized, accessed, and manipulated through a database management system. *Id.* For example, documents and folders can be represented as objects. *Id.* at 3:29–35. The hierarchically superior folder object pointers specify documents. *Id.* at 8:19–20.

2. Claim52

Enfish contends that Microsoft's motivation-to-combine arguments are insufficient. PO Resp. 39. We are not persuaded that Microsoft has set forth a sufficient articulated reason with a rational underpinning to support a determination of obviousness. Microsoft states that one of ordinary skill in the art would have been motivated to combine the folder and document objects of Smith '162 with the objects disclosed in Visual Basic for the benefits of providing varying data architectures in the database. Pet. 53–54 (citing Ex. 1119 ¶¶ 413–419). Microsoft, however, relies on Dr. Hosking, who states that one of ordinary skill in the art would have been motivated to combine Visual Basic and Smith '162 because they "address the same technical issues and disclose closely related subject matters." Ex. 1119 ¶ 419. Although Dr. Hosking mentions that Smith '162 relates to managing documents and folders, he does not testify persuasively regarding the benefits of providing varying data architectures in connection with combining Visual Basic and Smith '162. Microsoft does not offer persuasive additional contentions or evidence supporting the asserted combination of Visual Basic and Smith '162.

For the foregoing reasons, Microsoft has not shown that claim 52 of the '775 patent is unpatentable as obvious.

E. Secondary Considerations

We note that factual inquiries for an obviousness determination include secondary considerations based on evaluation and crediting of objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966). Notwithstanding what the teachings of the prior art would have suggested to one with ordinary skill in the art at the time of the

invention, the totality of the evidence submitted, including objective evidence of nonobviousness, may lead to a conclusion that the claimed invention would not have been obvious to one with ordinary skill in the art. *In re Piasecki*, 745 F.2d 1468, 1471–72 (Fed. Cir. 1984). However, to accord substantial weight to objective evidence requires the finding of a nexus between the evidence and the merits of the claimed invention. *In re GPAC Inc.*, 57 F.3d 1573, 1580 (Fed. Cir. 1995); *see also In re Huang*, 100 F.3d 135, 140 (Fed. Cir. 1996) ("success is relevant in the obviousness context only if there is proof that the sales were a direct result of the unique characteristics of the claimed invention.").

Enfish contends that the claimed invention received industry accolades, including praise from Microsoft, satisfied a long-felt need, resulted in success where others had failed, as well as commercial success. PO Resp. 42–47. Enfish points to features of claim 17, which Enfish contends are the features of the claimed invention that resulted in the objective indicia of success to which Enfish refers. *Id.* at 43–44. Claim 17 of the '775 patent, however, is challenged on the basis that it is anticipated by Visual Basic. Secondary considerations do not weigh into determinations regarding anticipation. *Cohesive Techs., Inc. v. Waters Corp.*, 543 F.3d 1351, 1364 (Fed. Cir. 2008).

Because we conclude that Microsoft has not shown that claims 51–53 of the '775 patent are unpatentable for other reasons, the only claims that we need to evaluate are dependent claims 49 and 50 of the '775 patent. Microsoft asserts that claims 49 and 50 are obvious over the combination of Visual Basic with a reference that teaches referencing objects through two-way pointers. Pet. 49 (citing Ex. 1106, 7:29–35). Enfish has not shown a

nexus between any of the accolades or successes it says occurred and the use of two-way pointers.

Enfish additionally submits evidence that Enfish states shows that Microsoft failed at developing a suitable search engine. PO Resp. 47–49. We are not persuaded by this argument. The statements submitted by Enfish are not tied sufficiently to any claim at issue in this proceeding. Instead, the statements broadly refer to search engines. None of the statements reference two-way pointers.

Enfish further submits evidence that Enfish alleges shows commercial success. PO Resp. 49. In addition to the shortcomings discussed above, the proffered evidence is further deficient for the following reasons. Enfish contends that the evidence supports that its product was "well received," 75,000 users downloaded Enfish's software, and the functionality of the software resulted in a collaborative effort. *See* PO Resp. 49 (citing Ex. 2024; Ex. 2025, 2; Ex. 2030). Enfish's evidence, however, does not indicate that users paid for or actually used the downloaded software. The evidence also does not indicate how the number of downloads illustrates commercial acceptance, for example, as compared to downloads of other software at the time. Additionally, a planned collaborative effort does not indicate the results of the collaboration. Enfish's evidence does not establish commercial acceptance or financial success. *See In re Fielder*, 471 F.2d 640, 644 (CCPA 1973). Thus, Enfish's evidence is not sufficient to show commercial success.

We, therefore, determine that feature of pointers would have been a predictable variation within the technical grasp of a person of ordinary skill in the art, and are not persuaded by the objective evidence that claims 49 and

50 of the '775 patent are not obvious.

F. Motion to Correct Patent Owner Response

After institution of trial, Enfish timely filed a Patent Owner Response (Paper 24), along with the Jagadish Declaration (Ex. 2007). On September 16, 2014, Enfish filed an unopposed motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration, which Enfish contends correct only typographical errors and erroneous citations. Paper 38. We grant Enfish's September 16, 2014 request.

G. Joint Stipulation

On November 14, 2014, the parties filed a joint stipulation requesting that we expunge confidential versions of exhibits 2049–2058 and 2060–2065. Paper 55. The parties contend that Microsoft withdraws its motion to seal (Paper 28) provided that we expunge the confidential versions. Paper 55. Microsoft agrees that the sealed version of Exhibit 2059 may be unsealed. *Id.* We hereby grant the motion and expunge only confidential versions of exhibits 2049–2058 and 2060–2065.

H. Motion to Exclude

On November 3, 2014, Microsoft filed a motion to exclude Exhibit 2071, the Declaration of Dr. Sharad Mehrotra ("Mehrotra Declaration"), and two paragraphs of the Declaration of Louise Wannier ("Wannier Declaration," Exhibit 2077 ¶¶ 32, 33). Paper 48.

Regarding the Mehrotra Declaration, we agree with Microsoft's assertion that Dr. Mehrotra provides only conclusory opinions and, therefore, we give his Declaration no weight and do not rely on it in this Decision. 37 C.F.R. § 42.65(a). Because Microsoft has not argued

persuasively any other reason to exclude the Mehrotra Declaration, we dismiss Microsoft's request to exclude it as moot.

Regarding the Wannier Declaration, we disagree with Microsoft that "Patent Owner has no basis to file the Wannier Declaration as supplemental evidence because Microsoft has not moved to exclude the Armon Declaration." Paper 48, 4–5. Patent Owner is entitled to submit supplemental evidence in response to Microsoft's objection. 37 C.F.R. § 42.64(b)(2). Microsoft further contends that the Wannier Declaration inserts untimely, conclusory, and improper technical opinions. Paper 48, 5. Patent Owner contends that paragraphs 32 and 33 do not exceed the scope because they are submitted to support admissibility. Paper 58, 3–4. We agree with Microsoft that the Wannier Declaration provides conclusory technical opinions, and, therefore, give the technical opinions in paragraphs 32 and 33 of her Declaration no weight. Because Microsoft has not argued persuasively any other reason to exclude paragraphs 32 and 33 of the Wannier Declaration, we dismiss Microsoft's request to exclude it as moot.

IV. CONCLUSION

We conclude that Microsoft has demonstrated by a preponderance of the evidence that (1) claims 47 and 48 of the '775 patent are anticipated by Visual Basic and (2) claims 49 and 50 of the '775 patent are obvious over the combination of Visual Basic and Jensen.

We further conclude that Microsoft has not shown that claims 51–53 of the '775 patent are unpatentable as obvious. In addition, we terminate this proceeding with respect to claims 17–23 under 37 C.F.R. § 42.72.

This is a final written decision of the Board under 35 U.S.C. § 318(a). Parties to the proceeding seeking judicial review of this decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

V. ORDER

For the reasons given, it is

ORDERED that claims 47–50 of U.S. Patent No. 6,163,775 are determined by a preponderance of the evidence to be unpatentable;

FURTHER ORDERED that this proceeding is TERMINATED, under 37 C.F.R. § 42.72, with respect to claims 17–23;

FURTHER ORDERED Enfish's motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration (Paper 38) is GRANTED;

FURTHER ORDERED that Microsoft's motion to exclude (Paper 48) is DISMISSED;

FURTHER ORDERED that confidential versions of Exhibits 2049–2058 and 2060–2065 are expunged;

FURTHER ORDERED Microsoft's motion to seal is DISMISSED; and

FURTHER ORDERED that because this is a final written decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

For PETITIONER:

Chun M. Ng
Amy E. Simpson
Chad Campbell
Theodore H. Wimsatt
PERKINS COIE LLP
CNg@perkinscoie.com
ASimpson@perkinscoie.com
CCampbell@perkinscoie.com
TWimsatt@perkinscoie.com

For PATENT OWNER:

Frank Pietrantonio
Orion Armon
COOLEY LLP
fpietrantonio@cooley.com
oarmon@cooley.com
zpatdcdocketing@cooley.com

PTAB Decision in IPR2013-00561

Trials@uspto.gov
571.272.7822

Paper 63
Entered: March 2, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

ENFISH, LLC,
Patent Owner.

Case IPR2013-00561
Patent 6,163,775

Before THOMAS L. GIANNETTI, BRYAN F. MOORE, and BARBARA A. PARVIS, *Administrative Patent Judges.*

PARVIS, *Administrative Patent Judge.*

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.73

I. INTRODUCTION

A. Background

Microsoft Corporation ("Microsoft") filed a petition to institute an *inter partes* review of claims 16, 24–30, 46, and 54–60 of U.S. Patent No. 6,163,775 ("the '775 patent"). Paper 1 ("Pet."). We instituted trial for claims 16, 24–26, 30, 46, 54–56, and 60 of the '775 patent on certain grounds of unpatentability alleged in the Petition. Paper 13 ("Decision to Institute" or "Inst. Dec.").

After institution of trial, Patent Owner, Enfish, LLC ("Enfish"), filed a Patent Owner Response, along with a Declaration by Dr. H.V. Jagadish ("Jagadish Declaration").

On September 16, 2014, Enfish filed an unopposed motion to correct typographical errors in both papers and filed therewith the corrected Patent Owner Response ("PO Resp.," Paper 40) and the corrected Jagadish Declaration (Ex. 2007). We grant Enfish's motion to correct both papers.[1] Microsoft filed a Reply ("Pet. Reply"). Paper 35.

A consolidated hearing for IPR2013-00559, IPR2013-00560, IPR2013-00561, IPR2013-00562, and IPR2013-00563 was held on December 3, 2014. The transcript of the consolidated hearing has been entered into the record. Paper 62 ("Tr.").

We have jurisdiction under 35 U.S.C. § 6(c). This final written decision is issued pursuant to 35 U.S.C. § 318(a).

[1] For ease of reference, throughout we refer to Enfish's corrected Patent Owner Response (Paper 40) and corrected Jagadish Declaration (Exhibit 2007), both filed on September 16, 2014.

Microsoft has shown by a preponderance of the evidence that claims 46 and 54 of the ’775 patent are unpatentable. Microsoft has not shown that claims 55, 56, and 60 of the ’775 patent are unpatentable. For the reasons discussed below, we are unable to reach a determination on the alleged grounds of unpatentability over prior art for claims 16, 24–26 and 30. Accordingly, we terminate this proceeding with respect to claims 16, 24–26 and 30 under 37 C.F.R. § 42.72.

B. Additional Proceedings

In addition to this petition, Microsoft has filed a petition challenging the patentability of claims 1–15, 17–23, 31–45, and 47–53 of the ’775 patent. *See Microsoft Corp. v. Enfish, LLC,* IPR2013-00559; IPR2013-00560. Microsoft indicates that claims of the ’775 patent have been asserted against Microsoft in *Enfish LLC v. Microsoft Corporation, et al.*, Case No. 12-cv-7360 MRP in the Central District of California (“California case”). Pet. 2. Microsoft further contends that a final judgment against Enfish has been entered in the California case. Ex. 1268.

Microsoft also has filed two petitions challenging the patentability of claims 1–60 of U.S. Patent No. 6,151,604 (“the ’604 patent”).[2] *See Microsoft Corp. v. Enfish, LLC*, IPR2013-00562; IPR2013-00563.

C. The ’775 Patent

The ’775 patent (Ex. 1201) is titled, “Method and Apparatus Configured According to a Logical Table Having Cell and Attributes Containing Address Segments,” and generally relates to a system and

[2] The ’604 patent and the ’775 patent both issued from continuations of Application No. 08/383,752, filed March 28, 1995, now U.S. Patent No. 5,729,730. Pet. 5.

method for data storage, manipulation, and retrieval in a logical table of a database. This table is a logical structure, not a physical structure, stored in the memory. Ex. 1201, 6:39–41.

Figure 3 is reproduced below:

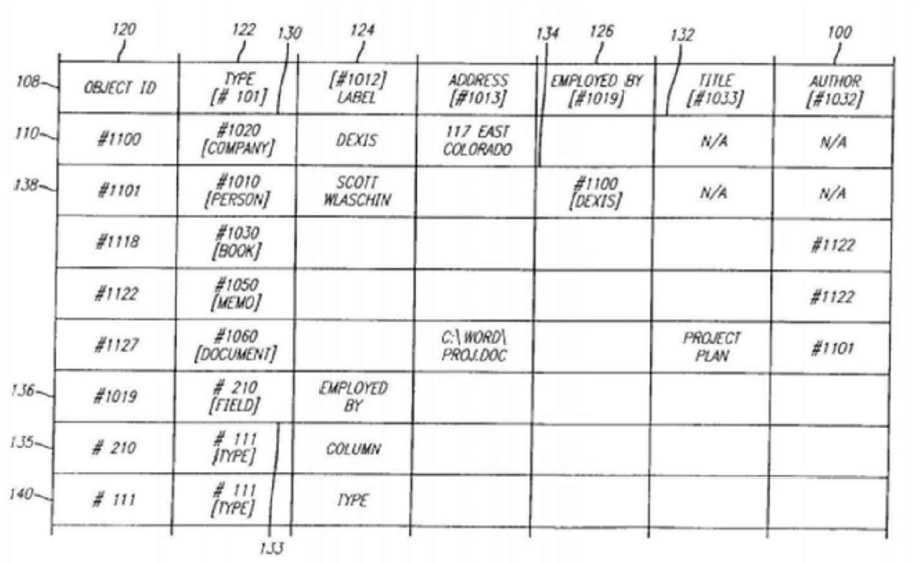


Figure 3 of the ’775 patent illustrates a structure of a logical table. Ex. 1201, 3:36–37. As depicted by Figure 3 of the ’775 patent, above, table 100 is defined by rows 108, 110, 138, 136, 135, and 140, and columns 120, 122, 124, and 126. *Id.* at Fig. 3. The intersection of a row and a column defines a cell in the table. *Id.* at 6:48–49. Each column corresponds to an attribute spanning various records. *Id.* at 6:46. An attribute is a single class description, such as an employer, denoted in column 126 of Figure 3, for example, by the text “Employed By.” *Id.* at 7:25–26.

Each row corresponds to a record spanning various attributes. Ex. 1201, 6:45–46. For example, row 110 corresponds to a company as shown in cell 130. *Id.* at 6:62–64.

D. Illustrative claims

Of the challenged claims, the independent claims are 16, 24, 46, and 54. Claims 25, 26, and 30 depend, directly or indirectly, from claim 24. Claims 55, 56, and 60 depend, directly or indirectly, from claim 54. Claims 46 and 54 illustrate the claimed subject matter and are reproduced below:

> 46. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of cells, each said cell having a first address segment and a second address segment;
>
> a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and
>
> a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record; wherein said OID's are variable length.
>
> 54. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:
>
> configuring said memory according to a logical table, said logical table including:
>
> a plurality of cells, each said cell having a first address segment and a second address segment, at least one of said cells includes a pointer to an index record;
>
> a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record; and

indexing data stored in said table.

E. The Prior Art References Supporting Alleged Unpatentability

Microsoft relies on the following references:

MICROSOFT® VISUAL BASIC™ PROGRAMMING SYSTEM FOR WINDOWS™ VERSION 3.0 (1993) ("Visual Basic," Ex. 1204).

American National Standard for Information Systems – Database Language – SQL (ANSI INCITS 135-1992 (R1998) ("SQL-92," Ex. 1205).

GERARD SALTON & MICHAEL J. MCGILL, INTRODUCTION TO MODERN INFORMATION RETRIEVAL (1983) ("Salton," Ex. 1207).

Sudarshan Chawathe et al., *The TSIMMIS Project: Integration of Heterogeneous Information Sources* (Dep't Computer Sci. Stanford Univ.) ("Chawathe," Ex. 1216).

F. The Pending Grounds of Unpatentability

The following chart summarizes Microsoft's patentablilty challenges:

Reference[s]	Basis	Claims Challenged
Visual Basic	§ 102(b)	24 and 54
SQL-92 and Chawathe	§ 103	16 and 46
Visual Basic and Salton	§ 103	25, 26, 30, 55, 56, and 60

In support of the above-referenced grounds of unpatentability, Microsoft relies on the Declaration of Dr. Hosking (Ex. 1219, "Hosking Declaration").

II. CLAIM CONSTRUCTION

A. Legal Standard

In an *inter partes* review, claim terms in an unexpired patent are interpreted according to their broadest reasonable construction in light of the specification of the patent in which they appear. 37 C.F.R. § 42.100(b); *see also In re Cuozzo Speed Techs., LLC.*, No. 14-01301, slip op. at 16, 19 (Fed. Cir. Feb. 4, 2015) ("Congress implicitly adopted the broadest reasonable interpretation standard in enacting the AIA," and "the standard was properly adopted by PTO regulation."). Under the broadest reasonable construction standard, claims are to be given their broadest reasonable interpretation consistent with the specification, and the claim language should be read in light of the specification as it would be interpreted by one of ordinary skill in the art. *In re Am. Acad. of Sci. Tech. Ctr.*, 367 F.3d 1359, 1364 (Fed. Cir. 2004). Also, we must be careful not to read a particular embodiment appearing in the written description into the claim, if the claim language is broader than the embodiment. *See In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993) ("[L]imitations are not to be read into the claims from the specification.").

B. Overview of the Parties' Positions

Microsoft contends that this case involves a computer-implemented invention. Pet. 6. Microsoft also contends that the challenged independent claims invoke means-plus-function claiming, but the '775 patent fails to disclose an algorithm, which is an issue under § 112, sixth paragraph, that

cannot be addressed in this proceeding. Pet. 11. In the Decision to Institute, we invited Enfish to direct us to the specific portions of the specification that clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. Inst. Dec. 11. For the reasons discussed below, we determine that Enfish in its response does not identify sufficient corresponding structure, as required under 35 U.S.C. § 112, sixth paragraph, for the "means for configuring said memory according to a logical table," recited in claims 16 and 24.

Regarding terms in the method claims, in the Decision to Institute, we provided constructions for "logical table," "object identification number," and "attribute set" which are shown in the table below. Inst. Dec. 11–13. We also determined that no express construction is needed for the following terms: "memory," "a cell having a first address segment and a second address segment," and "records." Inst. Dec. 15.

Claim Term or Phrase	Construction in the Decision to Institute
"logical table"	"[W]e construe the term "table" to mean: 'a structure of a database comprising rows and columns.'" Inst. Dec. 12. "We determine no express construction of 'logical' is needed for this decision." *Id.*
"object identification number"	"[W]e construe 'object identification number' in light of the specification to mean: 'an array of bits that define an object.'" *Id.* at 13.
"attribute set"	"[A] name or characteristic corresponding to a column." *Id.*

Enfish contends that our construction for "object identification number" is incomplete. PO Resp. 16. We evaluate Enfish's contention

below. Enfish also provides a proposed construction for "external documents," recited in claims 30 and 60. PO Resp. 22. Because we determine that Microsoft has not met its burden with respect to these claims for other reasons, no construction of this term for the purposes of this Decision is necessary.

We discern no reason, based on the complete record now before us, to change our constructions in the Decision to Institute of "logical table," "object identification number," and "attribute set."

C. Analysis of the Parties' Claim Construction Positions

1. Means for configuring said memory according to a logical table

Independent claims 16 and 24 include the limitation "means for configuring said memory according to a logical table." In the Decision to Institute, we agreed with Microsoft that under the broadest reasonable interpretation, the function for the means for configuring is "configuring memory according to a logical table." Inst. Dec. 9. Additionally, we considered the corresponding structure for the recited function as including a general purpose computer. *Id.* at 10. Enfish does not challenge persuasively either of these determinations; however, Enfish identifies portions of the specification that Enfish contends provide algorithmic support for the recited function. PO Resp. 23–25. In particular, Enfish contends that the '775 patent discloses a four-step algorithm that is linked to the recited function of configuring memory according to a logical table. *Id.*

"[T]he corresponding structure for a § 112 ¶ 6 claim for a computer-implemented function is the algorithm disclosed in the specification." *Aristocrat Techs. Austl. Party Ltd. vs. Int'l Game Tech.*, 521 F.3d 1328, 1333 (Fed. Cir. 2008) (quoting *Harris Corp. v. Ericsson Inc.*, 417 F.3d 1241,

1249 (Fed. Cir. 2005)). Additionally, specific portions of the specification must clearly link or associate a computer program or algorithm to the function corresponding to the claimed means. *See Medical Inst. & Diag. Corp. v. Elektra AB*, 344 F.3d 1205, 1211 (Fed. Cir. 2003). For the reasons set forth below, we conclude that the four steps and other '775 patent specification portions identified by Enfish do not describe an algorithm for the recited function of "configuring memory according to a logical table." We further conclude that Enfish has not clearly linked or associated the recited function to the four steps or the portions of the '775 patent specification that Enfish identifies.

For algorithmic support, Enfish identifies disparate excerpts of the '775 patent specification, which do not link or associate clearly a computer program or algorithm to the function corresponding to the claimed means for configuring said memory according to a logical table. For example, the first step, "[c]reate, in a computer memory, a logical table" (PO Resp. 23) appears to be similar to the recited function of configuring memory according to a logical table, but the first step is not found in the '775 patent specification. Additionally, none of the portions of the '775 patent specification that Enfish cites for this first step provide an algorithm or computer program for performing the recited function of "configuring memory according to a logical table" or clearly link or associate any algorithm or program to this recited function. PO Resp. 24 (citing Ex. 1201, Abstract, 2:59–63, 6:38–46, Figs. 1, 3, 9). These portions describe that an already-formed table has rows and columns, without describing how memory is configured to create a logical table having rows and columns. Ex. 1201, 2:59–63. One portion states that memories "need not store" the

table contiguously, but fails to describe an algorithm or computer program for configuring memory such that a logical table is not stored contiguously. *Id.* at 6:38–46.

Enfish's three remaining steps and the other '775 patent specification portions identified by Enfish fail to remedy these deficiencies. The second step is: "[a]ssign each row and column an object identification number (OID) that, when stored as data, can act as a pointer to the associated row or column." PO Resp. 24. One of the portions of the '775 patent specification cited by Enfish for the second step indicates "the system must generate a unique OID when columns and rows are formed." Ex. 1201, 8:15–16. The remainder of the identified portions relate to assigning an OID or indicate that an OID "may be used" as a pointer, without describing an algorithm for forming columns and rows of a table or showing how assigning an OID relates to steps for forming columns and rows. Ex. 1201, Abstract, 2:60–61, 6:50–57, 7:1–2, 8:18–60, Figs. 3, 4. Additionally, the identified portions discuss assigning a numeric value to an OID in the form of a bit array, but fail to describe how to configure memory such that an OID may be used as a pointer. *Id.*

The third step is a feature of claim 1 (PO Resp. 10). That claim is not challenged in this proceeding. Furthermore, the portions of the '775 patent specification cited by Patent Owner for the third step (PO Resp. 25 (citing Ex. 1201, Abstract, 2:66–3:2, 6:47–48, 7:25–31, Fig. 3)) indicate that a table has a row that corresponds to columns, such as a header row, which is a generic feature of an already-formed table. These '775 patent specification portions identified by Enfish (*id.*), however, do not describe how to form a table with this feature or link or associate this feature to the recited function.

The fourth step, i.e., storing and accessing data in cells (PO Resp. 25), is performed in an already-configured table and, therefore, occurs after the recited function of configuring a table has occurred. Nonetheless, the '775 patent specification portions identified by Enfish (*id.* (citing Ex. 1201, Abstract, 2:63–66, 6:48–49, 6:61–62, 7:9–10, 7:23–24, 11:52–62, Fig. 10)) suffer from the same deficiencies noted above.

We conclude that the four steps and '775 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table. The portions of the '775 patent specification identified by Enfish describe an already-formed table having generic characteristics, such as columns, rows, identifiers, and a header row. *See, e.g.*, Ex. 1201, Fig. 3. The description, however, does not disclose any algorithm or computer program for forming this table. Additionally, the description identified by Enfish is not linked or associated clearly with the recited function of "configuring memory according to a logical table."

Enfish also relies on Dr. Jagadish's Declaration to show that the '775 patent specification provides an algorithm and clearly links the algorithm to the recited function. PO Resp. 23–25 (citing Ex. 2007 ¶¶ 68–76). The Jagadish Declaration, however, relies on similar portions of the '775 patent specification to those cited in Patent Owner's Response. For the reasons given, the Jagadish Declaration does not support Enfish's assertion. Dr. Jagadish further asserts, "[i]t is also my opinion that one of ordinary skill in the art would understand how to implement those algorithm steps using techniques and resources that were available at the time the '775 Patent was filed." Ex. 2007 ¶ 69. Enfish, however, cannot rely on the knowledge of one skilled in the art to address the deficiencies noted above. *See Function*

Media, LLC v. Google Inc., 708 F.3d 1310, 1319 (Fed. Cir. 2013) ("Having failed to provide any disclosure of the structure . . . FM cannot rely on the knowledge of one skilled in the art to fill in the gaps.")

Because we conclude that the four steps and '775 patent specification portions identified by Enfish do not describe an algorithm for configuring memory in accordance with the claimed table, we terminate this proceeding with respect to the claims that recite this means-plus-function limitation. As explained in *BlackBerry Corporation v. Mobile Media Ideas, LLC*, Case IPR2013-00036 (PTAB Mar. 7, 2014) (Paper 65), the specification must disclose enough of a specific algorithm to provide the necessary structure under § 112, sixth paragraph. In the circumstance when the specification of the challenged patent lacks sufficient disclosure of structure under 35 U.S.C. § 112, sixth paragraph, the scope of the claims cannot be determined without speculation and, consequently, the differences between the claimed invention and the prior art cannot be ascertained. *Id.* For the reasons given, we determine that independent claims 16 and 24 are not amenable to construction and, thus, we terminate this proceeding with respect to claims 16 and 24 under 37 C.F.R. § 42.72. Because claims 25, 26, and 30 depend, directly or indirectly, from claim 24, we also terminate this proceeding with respect to these claims under 37 C.F.R. § 42.72.

2. *Other Means-Plus-Function Terms*

Because we determine that this proceeding should be terminated as to claims 25, 26, and 30 for the reasons discussed above, we need not construe other means-plus-function terms appearing in those claims for the purposes of this Decision.

3. Object Identification Number (OID)

Independent claims 46 and 54 recite "object identification number (OID)" and "OID." In the Decision to Institute, we construed "object identification number" in light of the specification to mean: "an array of bits that define an object." Inst. Dec. 13 (citing Ex. 1201, 3:38–40, 8:44–46).

Enfish "agrees in principle" with our construction, but contends that the construction is incomplete. PO Resp. 16. Enfish contends that "OID" should be defined further as "a unique, immutable, and system-generated value that identifies an object." *Id.* Microsoft contends that Enfish seeks to read into the claims extraneous limitations that are unsupported by either intrinsic or extrinsic evidence. Pet. Reply 2. Microsoft additionally contends that Enfish's proposed construction does not provide an appropriate context for the additional limitations. *See, e.g.*, Pet. Reply 3.

At the heart of Enfish's contention is an assertion that an OID will not function properly if it is not unique, immutable, and system generated. PO Resp. 16–22. For example, Enfish, relying on Dr. Jagadish, contends that an OID must be a unique value because "if an OID were not unique the database would be non-functional because objects could not be reliably retrieved." PO Resp. 17 (citing Ex. 2007 ¶ 51). Dr. Jagadish cites to a portion of the specification stating that the OID is used for "exact retrieval" and states that without uniqueness, retrieval using an OID will result in more than one object. Ex. 2007 ¶ 51 (citing Ex. 1201, 1:60–64).

We determine that Enfish's proposed construction of OID introduces requirements beyond what is needed for an OID to identify data and retrieve data from memory. Enfish, for example, contends that "an OID is unique among all rows and columns of the logical table." PO Resp. 29.

Additionally, claims 46 and 54 recite that an OID is included "to identify" each attribute set and each record, but do not specifically recite using an OID to retrieve data from memory.

Microsoft advances additional contentions regarding why Enfish's proposed construction should not be adopted. For example, Microsoft contends that Enfish's "unique" OID requirement would conflict with the patent, which describes a row in a table with an OID that has the same value as the OID of a column in the same table. Pet. Reply 3 (citing Ex. 1201, Fig. 3). Microsoft also contends that "immutable" should not be imported into the construction of OID because the term is absent from the specification, claims, and file history, and the specification is devoid of disclosure relating to immutability of an OID. Pet. Reply 3–4. As further support, Dr. Hosking states that an OID may change in certain circumstances and still retrieve information reliably. Ex. 1243 ¶¶ 21–25.

We determine that adopting Enfish's proposed construction without further context would create ambiguity. Additionally, limitations are not to be read into the claims from embodiments in the specification. *See In re Van Geuns*, 988 F.2d at 1184. Furthermore, we agree with Microsoft that Enfish's reliance on extrinsic evidence is misplaced. Enfish, for example, relies on excerpts of "text books" (PO Resp. 21) for its contention that an OID is "immutable," without showing sufficiently that the definition ascertained from these excerpts of extrinsic evidence is consistent with the definition that would be ascertained by reading the patent documents.

We, therefore, determine that the construction of "OID" in the Decision to Institute should not be changed.

III. ANALYSIS

A. Alleged Anticipation of Claim 54 by Visual Basic

For the reasons given below, after consideration of the arguments in Enfish's Patent Owner Response, and the evidence cited therein, we conclude that Microsoft has shown, by a preponderance of the evidence, that claim 54 is unpatentable as anticipated by Visual Basic.

1. Visual Basic

Visual Basic describes a programming system for the Windows™ operating system that enables programmers to create databases. Ex. 1204, PF 1, PG 453.[3] The programming system uses objects to represent tables of a database. *Id.* at PF 47. As described in Visual Basic, a table object is a logical representation of a physical table with records (rows) and fields (columns). *Id.* at LR 558.

Visual Basic also describes adding an index to a database, "[a]dding an index to your database can increase the speed with which you get access to information." Ex. 1204, PF 37. Visual Basic provides an example of an index with key fields Name, Last Name, City, and Zip Code. *Id.* at 38.

2. Claim 54

We have reviewed Microsoft's anticipation argument, supporting evidence, and the detailed claim analysis, which reads persuasively all elements of claim 54 onto the disclosure of Visual Basic. Pet. 24– 37 (citing Ex. 1204, LR 162, 185, 276, 280, 425, 532, 558, PF 1, 3, 18, 37–40, 47, 76,

---

[3] Citations in this decision to Exhibit 1204 refer to the volumes Programmer's Guide (PG), Language Reference (LR), and Professional Features (PF), respectively, as well as the page number within each volume.

80, PG 453, 471, 481, 482, 484, Ex. 1219 ¶¶ 67–93, 119–131). For example, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses configuring memory according to a logical table, as recited in claim 54. In the cited portions above, Visual Basic discloses storing data in a computer formatted as a table object, which is "a logical representation of a physical table." Ex. 1204, LR 558. The table object comprises "records (rows) and fields (columns)." *Id.*

As an additional example, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses each record and each attribute set including an OID, as recited in claim 54. In the cited portions above, Visual Basic describes assigning a primary key and that "[a] table's primary key is the determining factor when testing to see if the record is unique within the table." Ex. 1204, PF 39. According to Visual Basic "[Visual Basic] creates an index on the primary key of the table and uses it to find records and to create joins between tables." *Id.* Regarding each column including an OID, according to Visual Basic, "[e]ach Field in the Fields collection of a TableDef has a unique value in the OrdinalPosition property." *Id.* at LR 425.

Furthermore, we are persuaded that Microsoft has shown by a preponderance of the evidence that Visual Basic discloses indexing data stored in the table and a cell including a pointer to an index record, as recited in claim 54. In the cited portions above, Visual Basic discloses an index object used for creating indexes to "speed up the process of finding records." Ex. 1204, LR 276; *see also id.* at PF 37, 38, PG 482. Visual Basic also discloses a cell pointing to an index object. *Id.* at LR 280, PG 484.

Enfish contends that Microsoft has not met its burden of showing that

Visual Basic discloses OIDs. PO Resp. 29–34. Enfish's contentions are based on its proposed construction of OID, which we decline to adopt for the reasons given above.

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 54 of the '775 patent is anticipated by Visual Basic.

B. Obviousness of Claim 46 over SQL-92 and Chawathe

For the reasons given below, after consideration of Enfish's Patent Owner Response, and the evidence cited therein, we conclude that Microsoft has shown, by a preponderance of the evidence, that claim 46 is unpatentable as obvious over the combination of SQL-92 and Chawathe.

1. SQL-92

SQL-92 describes a standard specifying the syntax and semantics of the SQL database language. Ex. 1205, 1. In particular, SQL-92 describes a table object, which is a multi-set of rows. *Id.* at 12, 29; *see also id.* at 82 ("A <table name> identifies a table."). Additionally, the SQL-92 table object comprises columns. *Id.* at 12; *see also id.* at 82 ("A <column name> identifies a column.").

2. Chawathe

Chawathe describes a project for integrating heterogeneous information sources referred to as The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) project. Ex. 1216, 1. Chawathe states, for the TSIMMIS project, an object exchange model (OEM) is adopted. *Id.* Additionally, an SQL-like query language, OEM-QL, is adopted for requesting OEM objects. *Id.* at 2.

3. Claim 46

We have reviewed Microsoft's obviousness argument, supporting evidence, and the detailed claim analysis, which reads persuasively all elements of claim 46 onto the teachings of SQL-92 and Chawathe, taken together. Pet. 37–47 (citing Ex. 1205 1, 12, 29, 30, 82, 92, 156, 312–14, 316, 322, 477, 491, 497; Ex. 1216 2, 5, 6; Ex. 1219 ¶¶ 134–163, 385–386). For instance, we are persuaded that Microsoft has shown by a preponderance of the evidence that the combination of SQL-92 and Chawathe teaches configuring memory according to a logical table, as recited in claim 46. SQL-92 teaches "the descriptors of enabling objects (e.g., tables) are said to *include* the descriptors of enabled objects (e.g., columns or table constraints)." Ex. 1205, 12. Additionally, SQL-92 teaches "[a] table is a multiset of rows." *Id.* at 29.

Additionally, we determine that Microsoft has shown by a preponderance of the evidence that the combination of SQL-92 and Chawathe teaches each record and each attribute set including an OID, as recited in claim 46. For instance, SQL-92 teaches assigning a primary key that uniquely identifies a row in a table. Ex. 1205, 32–33, 497. SQL-92 also teaches that each column has an ordinal position. *Id.* at 29. Furthermore, Chawathe teaches a unifying object model, in which each object has a unique variable-length identifier referred to as "Object-ID." Ex. 1216, 5.

Enfish contends that Microsoft's motivation-to-combine arguments are insufficient. PO Resp. 34. We, however, are persuaded that Microsoft has set forth a sufficient articulated reason with a rational underpinning to support obviousness. *See KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 418 (2007) (citing *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006)). For

instance, Microsoft relies on Dr. Hosking's Declaration (Ex. 1219 ¶¶ 385–86), which states it would have been obvious to one of ordinary skill in the art to combine the techniques of SQL-92 and Chawathe to use an SQL-like language to query objects because Chawathe suggests combination with SQL-92 to enhance SQL-92 language capabilities. Ex. 1219 ¶ 386 (citing Ex. 1216, 2, 6). Chawathe states "OEM-QL is an SQL-like language extended to deal with labels and object nesting . . . ." Ex. 1216, 2; *see also id.* at 6 ("OEM-QL adapts existing SQL-like languages for object-oriented models . . . to OEM."). We credit Dr. Hosking's statement as it is consistent with the teachings of Chawathe (*see, e.g.*, Ex. 1216, 2. 6).

For the foregoing reasons, Microsoft has shown, by a preponderance of the evidence, that claim 46 of the '604 patent is unpatentable as obvious over the combination of SQL-92 and Chawathe.

C. *Obviousness of Claims 55, 56, and 60 in view of Visual Basic and Salton*

For the reasons given below, Microsoft has not shown that claims 55, 56, and 60 are unpatentable as obvious.

1. *Salton*

Salton describes information retrieval. Ex. 1207, 7. In particular, Salton describes speeding up a search for information by developing an index. *Id.* at 16. An index of Salton includes values for each key for records in a file. *Id.* at 17.

2. *Claims 55, 56, and 60*

Enfish contends that Microsoft's motivation-to-combine contentions are insufficient. PO Resp. 34–36. Dr. Jagadish states, "it is my opinion that

a POSITA would not have been motivated to combine [Visual Basic] and Salton as Petitioner proposes." Ex. 2007 ¶ 245. Dr. Jagadish provides various reasons including a lack of teachings in Visual Basic, difficulties described in Salton, as well as difficulties that would have been known to one of ordinary skill in the art. Ex. 2007 ¶¶ 245–246.

Microsoft discusses reasons for combining Visual Basic and Salton in three places in the Petition. First, Microsoft states that one of ordinary skill in the art would have been motivated to combine Salton with Visual Basic to employ the indexing and text analysis techniques taught by Salton in the object database systems of Visual Basic. Pet. 48 (citing Ex. 1219 ¶¶ 476–483). We are not persuaded that this is a sufficient rationale for combining Visual Basic and Salton, because Microsoft's citation does not provide adequate support. Microsoft cites only to Dr. Hosking, who states that Visual Basic and Salton are in the same field and provides a conclusion that one of ordinary skill in the art would have been motivated to combine Visual Basic and Salton. Ex. 1219 ¶ 479. Additionally, in its contentions regarding another challenged claim, Microsoft relies on Visual Basic for describing indexing. *See* Pet. 35 (Ex. 1204, PF 37, 38). In light of these other contentions, Microsoft does not explain sufficiently why one of ordinary skill in the art would have been motivated to employ Salton's indexing techniques with Visual Basic.

Second, Microsoft states, "[o]ne of ordinary skill in the art would combine Salton with [Visual Basic] to search for key words and create index records for those key words with pointers to the cells in [Visual Basic], because both Salton and [Visual Basic] discuss information search and retrieval using computer databases." Pet. 49–50 (citing Ex. 1219 ¶¶

457–483); *see also* Pet. 51 ("One of ordinary skill in the art would know to combine Salton and [Visual Basic] because both describe storing data in databases, allowing users to query the data, and using indexes to facilitate those queries"). Microsoft's contentions are insufficient because they are based on solely the references teaching the same techniques. Dr. Hosking states, "[o]ne having ordinary skill in the art would know to combine what's described in Salton with what's described in [Visual Basic] because they both deal with solving the problem of locating key words in a database." Ex. 1219 ¶ 468. Again, Dr. Hosking's conclusion is based solely on the references being in the same field. He fails to provide sufficient rational underpinning to explain why a person of ordinary skill would have combined the references.

Third, Microsoft states "[o]ne of ordinary skill in the art would know to add the weighting techniques described in Salton to [Visual Basic] to weigh key words and retrieve[] cells." Pet. 54 (citing Ex. 1219 ¶¶ 497–503). In the cited portions of the Hosking Declaration, Dr. Hosking states that "[o]ne having skill in the art would know to combine [Visual Basic] with the weigh[t]ing techniques of Salton because both have to do with providing access to data in a database through a search." Ex. 1219 ¶ 499. Although Dr. Hosking indicates that weighting increases the value of search results (*id.* ¶ 497), he does not indicate that increasing value using weighting is a reason for combining the technology of Salton with Visual Basic to arrive at the claims, which do not recite weighting.

As discussed above, Dr. Jagadish disagrees with Dr. Hosking. Ex. 2007 ¶¶ 245–246. In its Reply, Microsoft does not address Enfish's contentions or the testimony provided by Dr. Jagadish. Pet. Reply 11–12.

3. Conclusion

We determine that Dr. Hosking's statements are not a sufficient rationale for combining these references because he bases his conclusions on simple statements that the references are in the same field. Microsoft does not provide sufficient additional reasons, such as expert testimony that the combinations of prior art references are of familiar elements according to known methods that yield no more than predictable results. We, therefore, conclude that Microsoft has not demonstrated that the teachings would have been combined by a person of ordinary skill.

For the foregoing reasons, we determine that Microsoft has not set forth a sufficient articulated reason with a rational underpinning to support a showing of obviousness by a preponderance of the evidence. We, therefore, conclude that Microsoft has not shown that claims 55, 56, and 60 of the '775 patent are unpatentable as obvious.

D. Secondary Considerations

We note that factual inquiries for an obviousness determination include secondary considerations based on evaluation and crediting of objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17 (1966). Notwithstanding what the teachings of the prior art would have suggested to one with ordinary skill in the art at the time of the invention, the totality of the evidence submitted, including objective evidence of nonobviousness, may lead to a conclusion that the claimed invention would not have been obvious to one with ordinary skill in the art. *In re Piasecki*, 745 F.2d 1468, 1471–72 (Fed. Cir. 1984). However, to accord substantial weight to objective evidence requires the finding of a nexus between the evidence and the merits of the claimed invention. *In re*

GPAC Inc., 57 F.3d 1573, 1580 (Fed. Cir. 1995); *see also In re Huang*, 100 F.3d 135, 140 (Fed. Cir. 1996) ("success is relevant in the obviousness context only if there is proof that the sales were a direct result of the unique characteristics of the claimed invention.").

Enfish contends that the claimed invention received industry accolades, including praise from Microsoft, satisfied a long-felt need, resulted in success where others had failed, as well as commercial success. PO Resp. 37–42. Enfish points to features of claim 24, which Enfish contends are the features of the claimed invention that resulted in the objective indicia of success to which Enfish refers. *Id.* at 37–38. Claim 24 of the '775 patent, however, is challenged on the basis that it is anticipated by Visual Basic. Secondary considerations do not weigh into determinations regarding anticipation. *Cohesive Techs., Inc. v. Waters Corp.*, 543 F.3d 1351, 1364 (Fed. Cir. 2008).

Independent claim 46, which is challenged on the basis of obviousness, recites some of the same features as claim 24. We, however, determine that SQL-92 teaches all of the features of claim 46, including OIDs, except SQL-92 does not teach explicitly that certain OIDs have variable lengths. Microsoft contends that SQL-92 should be combined with Chawathe because Chawathe teaches each object having a unique variable-length identifier referred to as "Object-ID." Pet. 46 (citing Ex. 1216, 5). Enfish does not claim to have invented a variable length for an OID. Thus, Enfish has not shown a nexus between any of the accolades or successes it says occurred and the use of the claimed variable length OIDs.

Enfish additionally submits evidence that Enfish states shows that Microsoft failed at developing a suitable search engine. PO Resp. 42–44.

We are not persuaded by this argument. The statements submitted by Enfish are not tied sufficiently to any claim at issue in this proceeding. Instead, the statements broadly refer to search engines. None of the statements reference a variable length for an OID.

Enfish further submits evidence that Enfish alleges shows commercial success. PO Resp. 44. In addition to the shortcomings discussed above, the proffered evidence is further deficient for the following reasons. Enfish contends that the evidence supports that its product was "well received," 75,000 users downloaded Enfish's software, and the functionality of the software resulted in a collaborative effort. *See* PO Resp. 44 (citing Ex. 2024; Ex. 2025, 2; Ex. 2030). Enfish's evidence, however, does not indicate that users paid for or actually used the downloaded software. The evidence also does not indicate how the number of downloads illustrates commercial acceptance, for example, as compared to downloads of other software at the time. Additionally, a planned collaborative effort does not indicate the results of the collaboration. Enfish's evidence does not establish commercial acceptance or financial success. *See In re Fielder*, 471 F.2d 640, 644 (CCPA 1973). Thus, Enfish's evidence is not sufficient to show commercial success.

We, therefore, determine that the feature of variable length OIDs would have been a predictable variation within the technical grasp of a person of ordinary skill in the art, and are not persuaded by the objective evidence that claim 46 of the '775 patent is not obvious.

E. Motion to Correct Patent Owner Response

After institution of trial, Enfish timely filed a Patent Owner Response (Paper 25), along with the Jagadish Declaration (Ex. 2007). On September

16, 2014, Enfish filed an unopposed motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration, which Enfish contends correct only typographical errors and erroneous citations. Paper 38. We grant Enfish's September 16, 2014 request.

F. Joint Stipulation

On November 14, 2014, the parties filed a joint stipulation requesting that we expunge confidential versions of exhibits 2049–2058 and 2060–2065. Paper 55. The parties contend that Microsoft withdraws its motion to seal (Paper 28) provided that we expunge the confidential versions. Paper 55. Microsoft agrees that the sealed version of Exhibit 2059 may be unsealed. *Id.* We hereby grant the motion and expunge only confidential versions of exhibits 2049–2058 and 2060–2065.

G. Motion to Exclude

On November 3, 2014, Microsoft filed a motion to exclude Exhibit 2071, the Declaration of Dr. Sharad Mehrotra ("Mehrotra Declaration"), and two paragraphs of the Declaration of Louise Wannier ("Wannier Declaration," Exhibit 2077 ¶¶ 32, 33). Paper 48.

Regarding the Mehrotra Declaration, we agree with Microsoft's assertion that Dr. Mehrotra provides only conclusory opinions and, therefore, we give his Declaration no weight and do not rely on it in this Decision. 37 C.F.R. § 42.65(a). Because Microsoft has not argued persuasively any other reason to exclude the Mehrotra Declaration, we dismiss Microsoft's request to exclude it as moot.

Regarding the Wannier Declaration, we disagree with Microsoft that "Patent Owner has no basis to file the Wannier Declaration as supplemental

evidence because Microsoft has not moved to exclude the Armon Declaration." Paper 48, 4–5. Patent Owner is entitled to submit supplemental evidence in response to Microsoft's objection. 37 C.F.R. § 42.64(b)(2). Microsoft further contends that the Wannier Declaration inserts untimely, conclusory, and improper technical opinions. Paper 48, 5. Patent Owner contends that paragraphs 32 and 33 do not exceed the scope because they are submitted to support admissibility. Paper 58, 3–4. We agree with Microsoft that the Wannier Declaration provides conclusory technical opinions, and, therefore, give the technical opinions in paragraphs 32 and 33 of her Declaration no weight. Because Microsoft has not argued persuasively any other reason to exclude paragraphs 32 and 33 of the Wannier Declaration, we dismiss Microsoft's request to exclude it as moot.

IV. CONCLUSION

We conclude that Microsoft has demonstrated by a preponderance of the evidence that (1) claim 54 of the '775 patent is anticipated by Visual Basic and (2) claim 46 of the '775 patent is obvious over the combination of SQL-92 and Chawathe. We further conclude that Microsoft has not shown that claims 55, 56, and 60 of the '775 patent are unpatentable as obvious. In addition, we terminate this proceeding with respect to claims 16, 24–26 and 30 under 37 C.F.R. § 42.72. Claims 27–29 and 57–59 are not at issue in this trial.[4]

[4] In the Decision to Institute, we declined to institute an *inter partes* review of claims 27–29 and 57–59 because we were not persuaded that Petitioner had shown that there was a reasonable likelihood of prevailing on its challenges to these claims. Inst. Dec. 2, 24–25.

This is a final written decision of the Board under 35 U.S.C. § 318(a). Parties to the proceeding seeking judicial review of this decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

V. ORDER

For the reasons given, it is

ORDERED that claims 46 and 54 of U.S. Patent No. 6,163,775 are determined by a preponderance of the evidence to be unpatentable;

FURTHER ORDERED that this proceeding is TERMINATED, under 37 C.F.R. § 42.72, with respect to claims 16, 24–26 and 30;

FURTHER ORDERED Enfish's motion to file a second corrected Patent Owner Response and a corrected Jagadish Declaration (Paper 38) is GRANTED;

FURTHER ORDERED that Microsoft's motion to exclude (Paper 48) is DISMISSED;

FURTHER ORDERED that confidential versions of Exhibits 2049–2058 and 2060–2065 are expunged;

FURTHER ORDERED Microsoft's motion to seal is DISMISSED; and

FURTHER ORDERED that because this is a final written decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

For PETITIONER:

Chun M. Ng
Amy E. Simpson
Chad Campbell
Theodore H. Wimsatt
PERKINS COIE LLP
CNg@perkinscoie.com
ASimpson@perkinscoie.com
CCampbell@perkinscoie.com
TWimsatt@perkinscoie.com

For PATENT OWNER:

Frank Pietrantonio
Orion Armon
COOLEY LLP
fpietrantonio@cooley.com
oarmon@cooley.com
zpatdcdocketing@cooley.com

PTAB Rehearing Decision in IPR2013-00562

Trials@uspto.gov
571-272-7822

Paper 69
Entered: June 17, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

ENFISH, LLC,
Patent Owner.

Case IPR2013-00562
Patent 6,151,604

Before THOMAS L. GIANNETTI, BRYAN F. MOORE, and SCOTT A. DANIELS, *Administrative Patent Judges*.

DANIELS, *Administrative Patent Judge*.

DECISION
Denying Petitioner's Request for Rehearing
37 C.F.R. § 42.71

Microsoft filed a Motion for Rehearing (Paper 65, April 1, 2015 "Mot. Reh'g") of the Board's Final Written Decision (Paper 64, March 3, 2015, "Dec."), on one issue, namely, that Microsoft had failed to demonstrate that claim 32 of Enfish's Patent 6,151,604 (the "'604 patent") is unpatentable.

The burden of showing a decision should be modified is on the party challenging the decision. 37 C.F.R. § 42.71(d). Microsoft has not sustained its burden and, therefore, the Request is DENIED.

ANALYSIS

Standard of Review

Under 37 C.F.R. § 42.71(c), "[w]hen rehearing a decision on petition, a panel will review the decision for an abuse of discretion." Abuse of discretion occurs when a "decision was based on an erroneous conclusion of law or clearly erroneous factual findings, or . . . a clear error of judgment." *PPG Indus. Inc. v. Celanese Polymer Specialties Co.*, 840 F.2d 1565, 1567 (Fed. Cir. 1988) (citations omitted). In its request for rehearing, the dissatisfied party must identify the place in the record where it previously addressed each matter it submits for review. 37 C.F.R. § 42.71(d).

Whether SYS.TABLES Meets the Limitations of Claim 32

Microsoft contends that the Board overlooked Microsoft's argument and evidence that Chang's "SYS.TABLES *alone* contains the claimed indexing information from the SYS.INDEXES table." Mot. Reh'g 1. Specifically, Microsoft asserts that claim 32 is anticipated by Chang because "indexing information from the SYS.INDEXES table in Chang is incorporated into and stored within the SYS.TABLES packed descriptor (PD)," thus, "enabling determination of OIDs from text entry" as called for in claim 32. *Id.* at 3.

Claim 32 depends from independent claim 31 and reads as follows:

> 32. The method of claim 31 wherein said logical column information defines one of said logical columns to contain information for enabling determination of OIDs from text entry.

Microsoft argues specifically that the Board determined that the additional limitations of claim 32 are satisfied by the SYS.INDEXES table, and that, "[b]ecause the information in the SYS.INDEXES table is also stored in the SYS.TABLES table, claim 32 is anticipated by a single table in Chang -- the SYS.TABLES table." *Id.* at 5.

We are not convinced by this argument that anything has been overlooked or misapprehended. We determined in our Decision that all the limitations recited in the claims must be found in a single table. Dec. 24. We determined also that Microsoft's evidence and arguments were not persuasive to show that the packed description enabled "determination of OIDs from text entry," as recited in claim 32. Specifically, we stated that:

> We conclude that the evidence provided by Microsoft with respect to the packed description in SYS.TABLES storing column information, including primary key column definition information from the SYS.COLUMNS table is not persuasive because it does not explain how OID determination by text entry as recited in claim 32 would be conducted on such stored information and definitions in these particular tables, or even that SYS.TABLES or SYS.COLUMNS are indexed to provide such a search function.

Dec. 27 (citing Pet. 34–35). We then explained that the evidence was persuasive that Chang used the SYS.INDEX table in Figure 4, not the packed description, to determine what columns could be searched:

> Dr. Hosking describes that it is SYS.INDEX of Figure 4 that is necessary for Chang to determine what "columns are amenable

> to being searched." Ex. 1022 ¶ 184. Enfish argues persuasively that the only table Chang discloses for index searching is SYS.INDEX shown in Figure 4.

Id. (citing PO Resp. 42–43). Although Dr. Hosking states that SYS.INDEXES table information is "collected" in the packed description (Ex. 1022 ¶ 184) neither Dr. Hosking, nor Microsoft has pointed to any persuasive disclosure or evidence in Chang explaining where in the packed description such "information" includes an OID that could be determined based on a text entry query of the SYS.TABLES alone, or *how* the "information" from the indexing catalog contained in the packed description would have enabled text searching and return of an OID in the SYS.TABLES catalog alone. Indeed, during his deposition, Dr. Hosking could not confirm that any single table in Chang enabled determination of OID's from text entry:

> Q: My question in the eight paragraphs in the section for claim two, can you point to a single table -- a single logical table in which logical column information exists that defines one of the logical columns in that table to contain information for enabling determination of OIDs in that table from text entry?
>
> . . .
>
> A: Yeah. I think my answer was no.

Ex. 2003, 198:22–199:7.

In the Petition, Microsoft also alleged that the "[d]escriptions of the indexes of columns in a table is collected into the same packed descriptor, PD, as the other column defining information discussed with respect to claims 1 and 31." Pet. 35 (citing Ex. 1003 at Figs. 2, 6, 7, 10:34–11:4). In its Motion for Rehearing, Microsoft reiterates "that the indexing information

in SYS.INDEXES table 'is collected into the same packed descriptor, PD, as the other column defining information discussed with respect to claims 1 and 31,' and is thus incorporated into the SYS.TABLES table." Mot. Reh'g 8 (citing Pet. 35).

Microsoft's argument and evidence, however, never defines what "information" from the SYS.INDEXES is actually in the packed description that permits text searching for an OID. Mot. Reh'g 1 ("the SYS.TABLES table includes all of the pertinent information from the SYS.INDEXES table and thus anticipates claim 32"). Microsoft broadly assumes, without expressly stating, that the packed description includes *sufficient* indexing catalog information and functionality so that a text search query necessitating use of the SYS.INDEXES catalog would be conducted solely in the SYS.TABLES catalog itself. As shown in Chang's Figure 6, reproduced below, the information in the packed descriptions are column descriptions corresponding to a row in the SYS.INDEXES catalog. Ex. 1003, 10:50–54.

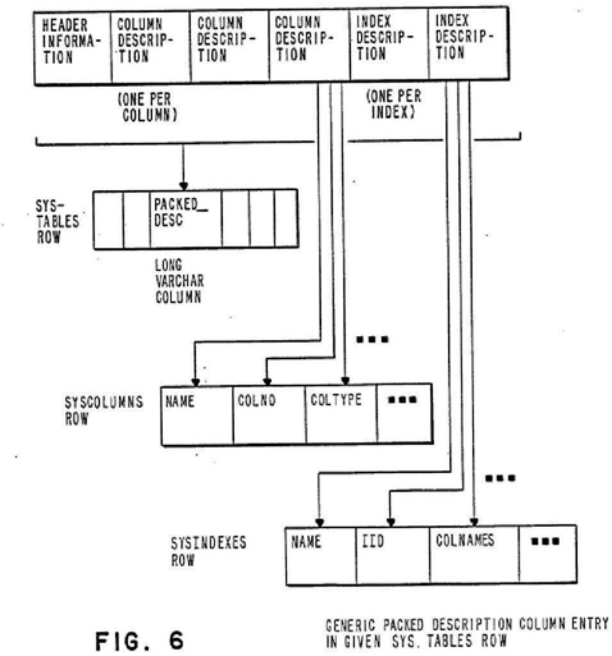


FIG. 6

Figure 6, above, illustrates a block diagram of the packed description showing an "INDEX DESCRIPTION" corresponding to a row entry in SYS.INDEXES. Chang's Figure 6, however, fails to disclose that the index description information permits OID determination from text entry; Figure 6 discloses only that it corresponds to additional information in the SYS.INDEXES catalog. Neither Microsoft nor Dr. Hosking explains how the index descriptions in the packed description, alone, enable determination of OIDs from text entry. Microsoft concludes that because the packed description contains descriptive attribute information relating to a row of a different table, the packed description can be searched for an OID by text entry. Mot. Reh'g 1, 3, 5–6. We are not persuaded by this argument because we find this conclusion to be without evidentiary support. Microsoft has not explained adequately, or provided persuasive evidence, as to how an OID can be determined by text entry from the packed description

in SYS.TABLES alone, or even without reference to SYS.INDEX table in Figure 4.

Microsoft further asserts that the Board accepted Microsoft's arguments and explanations relating to the incorporation of SYS.COLUMNS information, such as column numbers and column names, in the SYS.TABLES packed descriptor as anticipating claims 1 and 31. Mot. Reh'g 7. Therefore, Microsoft argues, "the indexing information in SYS.INDEXES table 'is collected into the same packed descriptor, PD, as the other column defining information discussed with respect to claims 1 and 31,' and is thus incorporated into the SYS.TABLES table." *Id.* at 8 (citing Pet. 35). This argument also is not persuasive because claims 1 and 31 do not contain the additional limitation defining "one of said logical columns to contain information for enabling determination of OIDs from text entry," as recited in claim 32. Although the index catalog description in the SYS.TABLES packed description potentially includes a column name or number, there is, again, no disclosure in Chang, or persuasive evidence offered by Microsoft, that such information enables "determination of OID's from text entry" in the SYS.TABLES alone.

Based on the evidence presented, we are not persuaded that Chang's packed description in the SYS.TABLES catalog anticipates the limitation in claim 32 requiring that "said attribute set information defines one of said attribute sets to contain information for enabling determination of OIDs from text entry."

Conclusion

Microsoft's request for rehearing is denied.

For PETITIONER:
Chung Ng
Amy E. Simpson
Chad Campbell
Theodore Wimsatt
PERKINS COIE LLP
CNg@perkinscoie.com
ASimpson@perkinscoie.com
CCampbell@perkinscoie.com
TWimsatt@perkinscoie.com

For PATENT OWNER:

Frank Pietrantonio
Orion Armon
COOLEY LLP
fpietrantonio@cooley.com
oarmon@cooley.com
zpatdcdocketing@cooley.com

PTAB Rehearing Decision in IPR2013-00559

Trials@uspto.gov
571-272-7822

Paper 70
Entered: June 17, 2015

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

ENFISH, LLC,
Patent Owner.

Case IPR2013-00559
Patent 6,163,775

Before THOMAS L. GIANNETTI, BRYAN F. MOORE, and SCOTT A. DANIELS, *Administrative Patent Judges.*

DANIELS, *Administrative Patent Judge.*

DECISION
Denying Petitioner's Request for Rehearing
37 C.F.R. § 42.71

Microsoft filed a Motion for Rehearing (Paper 66, April 1, 2015 "Mot. Reh'g") of the Board's Final Written Decision (Paper 65, March 3, 2015, "Dec."), on one issue, namely, that Microsoft had failed to demonstrate that claim 32 of Enfish's Patent 6,613,775 (the "'775 patent") is unpatentable.

The burden of showing a decision should be modified is on the party challenging the decision. 37 C.F.R. § 42.71(d). Microsoft has not sustained its burden and, therefore, the Request is DENIED.

ANALYSIS

Standard of Review

Under 37 C.F.R. § 42.71(c), "[w]hen rehearing a decision on petition, a panel will review the decision for an abuse of discretion." Abuse of discretion occurs when a "decision was based on an erroneous conclusion of law or clearly erroneous factual findings, or . . . a clear error of judgment." *PPG Indus. Inc. v. Celanese Polymer Specialties Co.*, 840 F.2d 1565, 1567 (Fed. Cir. 1988) (citations omitted). In its request for rehearing, the dissatisfied party must identify the place in the record where it previously addressed each matter it submits for review. 37 C.F.R. § 42.71(d).

Whether SYS.TABLES Meets the Limitations of Claim 32

Microsoft contends that the Board overlooked Microsoft's argument and evidence that Chang's "SYS.TABLES *alone* contains the claimed indexing information from the SYS.INDEXES table." Mot. Reh'g 1. Specifically, Microsoft asserts that claim 32 is anticipated by Chang because "indexing information from the SYS.INDEXES table in Chang is incorporated into and stored within the SYS.TABLES packed descriptor (PD)," thus, "enabling determination of OIDs from text entry" as called for in claim 32. *Id.* at 3.

Claim 32 depends from independent claim 31 and reads as follows:

> 32. The method of claim 31 wherein said attribute set information defines one of said attribute sets to contain information for enabling determination of OIDs from text entry.

Microsoft argues specifically that the Board determined that the additional limitations of claim 32 are satisfied by the SYS.INDEXES table, and that, "[b]ecause the information in the SYS.INDEXES table is also stored in the SYS.TABLES table; claim 32 is anticipated by a single table in Chang -- the SYS.TABLES table." *Id.* at 5–6.

We are not convinced by this argument that anything has been overlooked or misapprehended. We determined in our Decision that all the limitations recited in the claims must be found in a single table. Dec. 21–22, 24–25. We determined also that Microsoft's evidence and arguments were not persuasive to show that the packed description enabled "determination of OIDs from text entry," as recited in claim 32. Specifically, we stated that:

> The evidence provided by Microsoft with respect to the packed description in SYS.TABLES storing column information, including primary key column definition information from the SYS.COLUMNS table is not persuasive because it does not explain how OID determination, by text entry, as recited in claim 32 would be conducted on such stored information and definitions in these particular tables, or even that SYS.TABLES or SYS.COLUMNS are indexed to provide such a search function.

Dec. 25 (citing Pet. 39–40). We then explained that the evidence was persuasive that Chang used the SYS.INDEX table in Figure 4, not the packed description, to determine what columns could be searched:

> Dr. Hosking states that it is SYS.INDEX of Figure 4 that is necessary for Chang to determine what "columns are amenable

> to being searched." Ex. 100[2] ¶ 200. Enfish argues persuasively that the only table Chang discloses for index searching is SYS.INDEX shown in Figure 4.

Id. Although Dr. Hosking states that SYS.INDEXES table information is "collected" in the packed description (Ex. 1002 ¶ 200), neither Dr. Hosking, nor Microsoft has pointed to any persuasive disclosure or evidence in Chang explaining where in the packed description such "information" includes an OID that could be determined based on a text entry query of the SYS.TABLES alone, or *how* the "information" from the indexing catalog contained in the packed description would have enabled text searching and return of an OID in the SYS.TABLES catalog alone. Indeed, during his deposition, Dr. Hosking could not confirm that any single table in Chang enabled determination of OID's from text entry:

> Q: My question in the eight paragraphs in the section for claim two, can you point to a single table -- a single logical table in which logical column information exists that defines one of the logical columns in that table to contain information for enabling determination of OIDs in that table from text entry?
>
> . . .
>
> A: Yeah. I think my answer was no.

Ex. 2003, 198:22–199:7.

In the Petition, Microsoft also alleged that the "[d]escriptions of the indexes of columns in a table is collected into the same packed descriptor, PD, as the other column defining information discussed with respect to claims 1 and 31." Pet. 39 (citing Ex. 1003 at Figs. 2, 6, 7, 10:34–11:4). In its Motion for Rehearing, Microsoft reiterates "that the indexing information in SYS.INDEXES table "is collected into the same packed descriptor, PD, as the other column defining information discussed with respect to claims 1 and

31," and is thus incorporated into the SYS.TABLES table." Mot. Reh'g 8 (citing Pet. 39).

Microsoft's argument and evidence, however, never defines what "information" from the SYS.INDEXES is actually in the packed description that permits text searching for an OID. Mot. Reh'g 1 ("the SYS.TABLES table includes all of the pertinent information from the SYS.INDEXES table and thus anticipates claim 32"). Microsoft broadly assumes, without expressly stating, that the packed description includes *sufficient* indexing catalog information and functionality so that a text search query necessitating use of the SYS.INDEXES catalog would be conducted solely in the SYS.TABLES catalog itself. As shown in Chang's Figure 6, reproduced below, the information in the packed descriptions are column descriptions corresponding to a row in the SYS.INDEXES catalog. Ex. 1003, 10:50–54.

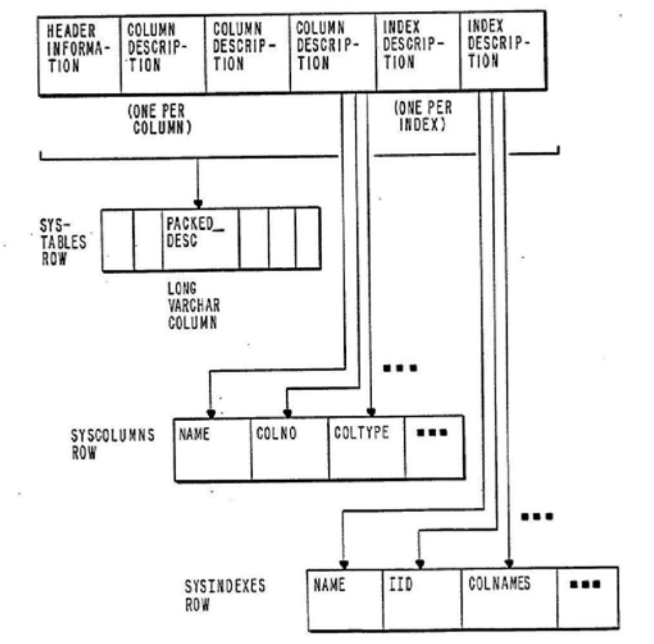


FIG. 6

Figure 6, above, illustrates a block diagram of the packed description showing an "INDEX DESCRIPTION" corresponding to a row entry in SYS.INDEXES. Chang's Figure 6, however, fails to disclose that the index description information permits OID determination from text entry; Figure 6 discloses only that it corresponds to additional information in the SYS.INDEXES catalog. Neither Microsoft nor Dr. Hosking explains how the index descriptions in the packed description, alone, enable determination of OIDs from text entry. Microsoft concludes that because the packed description contains descriptive attribute information relating to a row of a different table, the packed description can be searched for an OID by text entry. Mot. Reh'g 1, 3, 5–6. We are not persuaded by this argument because we find this conclusion to be without evidentiary support. Microsoft has not explained adequately, or provided persuasive evidence, as to how an OID can be determined by text entry from the packed description in SYS.TABLES alone, or even without reference to SYS.INDEX table, in Figure 4.

Microsoft further asserts that the Board accepted Microsoft's arguments and explanations relating to the incorporation of SYS.COLUMNS information, such as column numbers and column names, in the SYS.TABLES packed descriptor as anticipating claims 1 and 31. Mot. Reh'g 7. Therefore, Microsoft argues, "the indexing information in SYS.INDEXES table 'is collected into the same packed descriptor, PD, as the other column defining information discussed with respect to claims 1 and 31,' and is thus incorporated into the SYS.TABLES table." *Id.* at 8 (citing Pet. 39). This argument also is not persuasive because claims 1 and 31 do not contain the additional limitation defining "one of said attribute sets to

contain information for enabling determination of OIDs from text entry," as recited in claim 32. Although the index catalog description in the SYS.TABLES packed description potentially includes a column name or number, there is, again, no disclosure in Chang, or persuasive evidence offered by Microsoft, that such information enables "determination of OID's from text entry" in the SYS.TABLES alone.

Based on the evidence presented, we are not persuaded that Chang's packed description in the SYS.TABLES catalog anticipates the limitation in claim 32 requiring that "said attribute set information defines one of said attribute sets to contain information for enabling determination of OIDs from text entry."

Conclusion

Microsoft's request for rehearing is denied.

For PETITIONER:

Chung Ng
Amy E. Simpson
Chad Campbell
Theodore Wimsatt
PERKINS COIE LLP
CNg@perkinscoie.com
ASimpson@perkinscoie.com
CCampbell@perkinscoie.com
TWimsatt@perkinscoie.com

For PATENT OWNER:

Frank Pietrantonio
Orion Armon
COOLEY LLP
fpietrantonio@cooley.com
oarmon@cooley.com
zpatdcdocketing@cooley.com

’604 Patent

US006151604A

United States Patent [19]

Wlaschin et al.

[11] **Patent Number: 6,151,604**

[45] **Date of Patent: *Nov. 21, 2000**

[54] **METHOD AND APPARATUS FOR IMPROVED INFORMATION STORAGE AND RETRIEVAL SYSTEM**

[75] Inventors: **Scott Wlaschin**; **Robert M. Gordon**, both of Los Angeles; **Louise J. Wannier**, La Canada, all of Calif.; **Clay Gordon**, New York, N.Y.

[73] Assignee: **Dex Information Systems, Inc.**

[*] Notice: This patent is subject to a terminal disclaimer.

[21] Appl. No.: **09/035,510**

[22] Filed: **Mar. 5, 1998**

Related U.S. Application Data

[63] Continuation of application No. 08/383,752, Mar. 28, 1995, Pat. No. 5,729,730.

[51] **Int. Cl.**[7] **G06F 17/30**

[52] **U.S. Cl.** **707/100**; 707/102; 707/3

[58] **Field of Search** 707/3, 4, 100, 707/102

[56] **References Cited**

U.S. PATENT DOCUMENTS

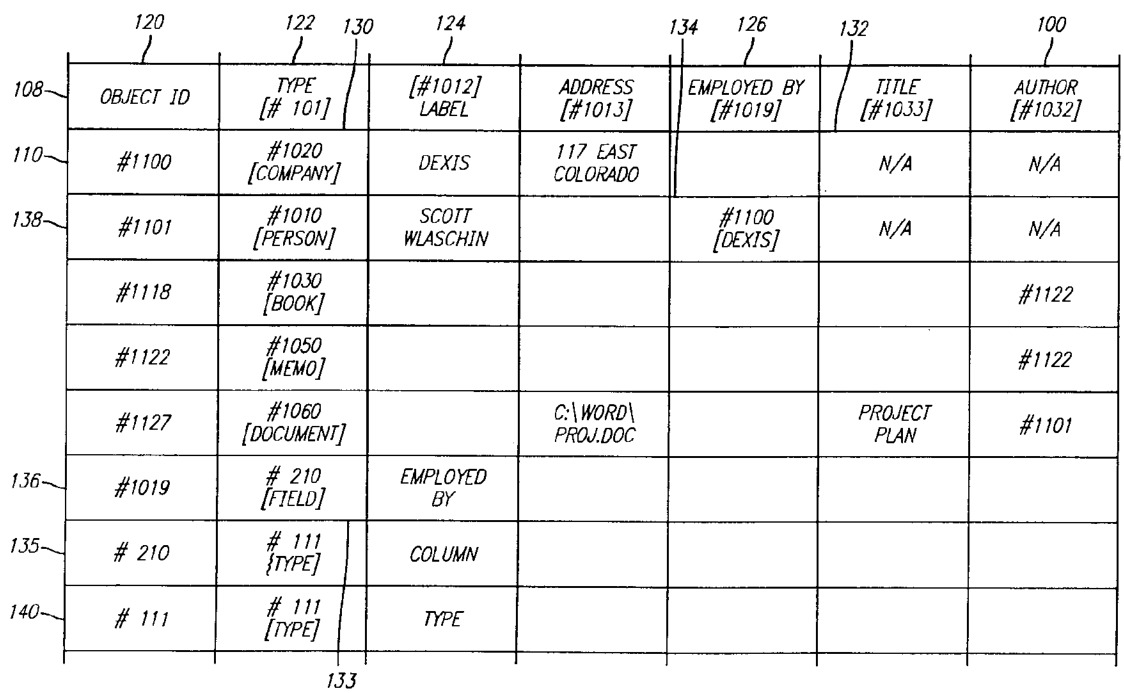
5,295,256	3/1994	Bapat	395/500
5,305,389	4/1994	Palmer	382/1
5,359,724	10/1994	Earle	395/425
5,375,237	12/1994	Tanaka et al.	395/650
5,421,012	5/1995	Khoyi et al.	395/650
5,459,860	10/1995	Burnett et al.	707/1
5,537,591	7/1996	Oka	707/4
5,537,633	7/1996	Suzuki et al.	707/3
5,553,218	9/1996	Li et al.	395/148
5,557,787	9/1996	Shin et al.	707/1
5,564,046	10/1996	Nemoto et al.	707/4
5,630,005	5/1997	Hoover et al.	707/3
5,729,730	3/1998	Wlaschin et al.	707/3

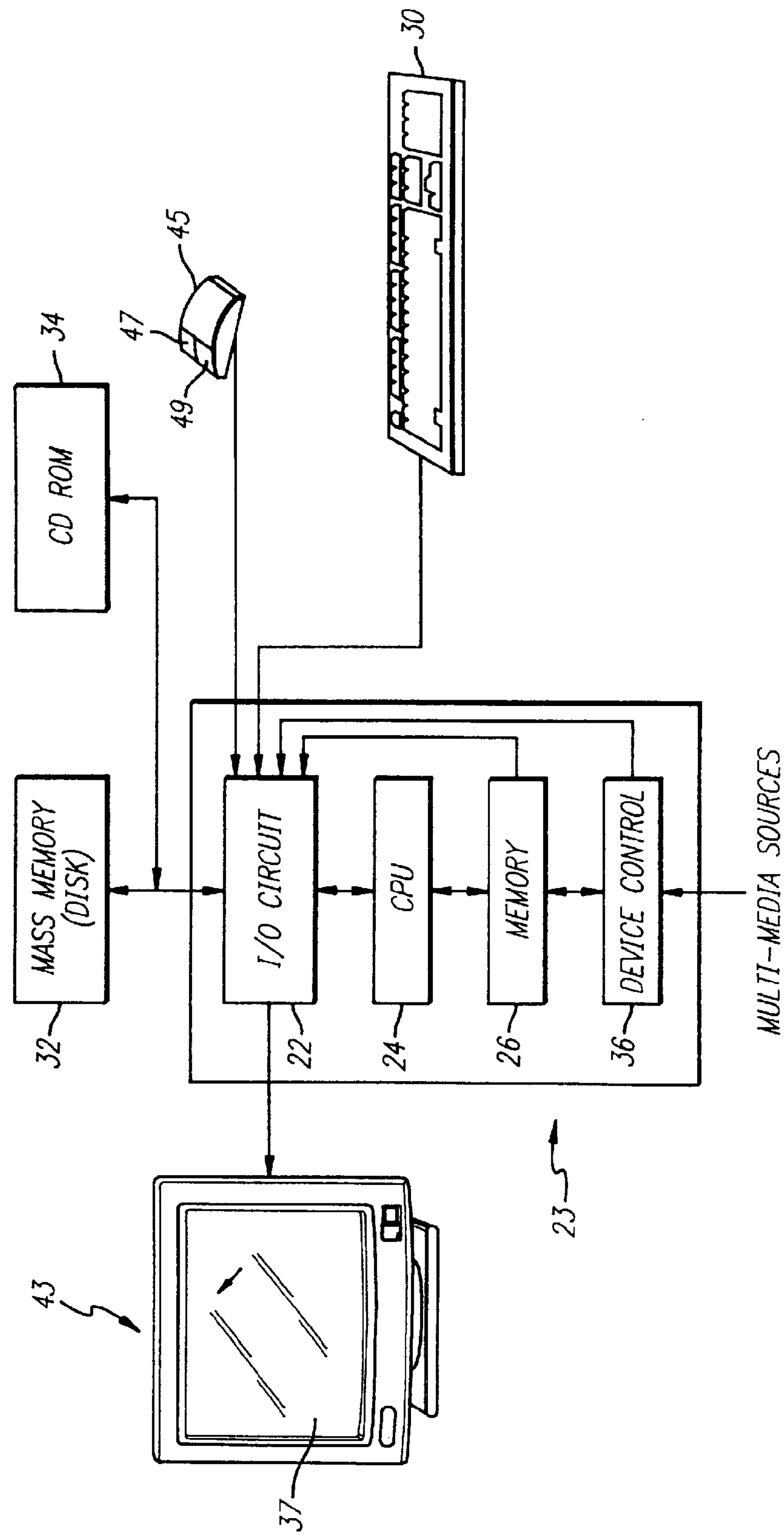
Primary Examiner—Thomas G. Black
Assistant Examiner—Frantz Coby
Attorney, Agent, or Firm—Morrison & Foerster LLP

[57] **ABSTRACT**

The information management and database system of the present invention comprises a flexible, self-referential table that stores data. The table of the present invention may store any type of data, both structured and unstructured, and provides an interface to other application programs. The table of the present invention comprises a plurality of rows and columns. Each row has an object identification number (OID) and each column also has an OID. A row corresponds to a record and a column corresponds to a field such that the intersection of a row and a column comprises a cell that may contain data for a particular record related to a particular field, a cell may also point to another record. To enhance searching and to provide for synchronization between columns, columns are entered as rows in the table and the record corresponding to a column contains various information about the column. The table includes an index structure for extended queries.

60 Claims, 17 Drawing Sheets





30
45
47
49
34
CD ROM
MASS MEMORY (DISK)
I/O CIRCUIT
CPU
MEMORY
DEVICE CONTROL
MULTI-MEDIA SOURCES
32
22
24
26
36
23
43
37

FIG. 1

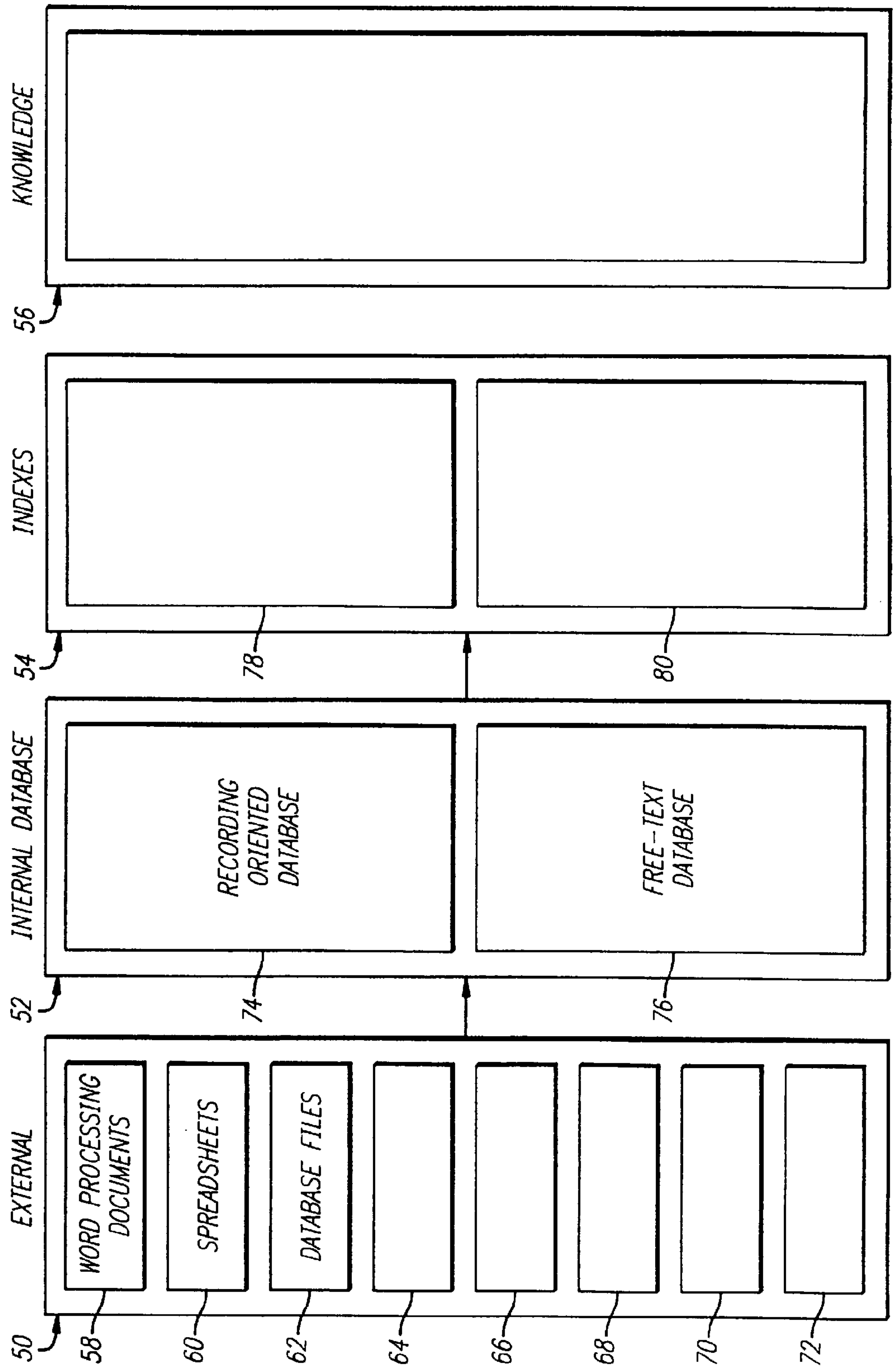

KNOWLEDGE
INDEXES
INTERNAL DATABASE
EXTERNAL
RECORDING ORIENTED DATABASE
FREE-TEXT DATABASE
WORD PROCESSING DOCUMENTS
SPREADSHEETS
DATABASE FILES
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80

FIG. 2

FIG. 3

OBJECT ID	TYPE [# 101]	[#1012] LABEL	ADDRESS [#1013]	EMPLOYED BY [#1019]	TITLE [#1033]	AUTHOR [#1032]
#1100	#1020 [COMPANY]	DEXIS	117 EAST COLORADO		N/A	N/A
#1101	#1010 [PERSON]	SCOTT WLASCHIN		#1100 [DEXIS]	N/A	N/A
#1118	#1030 [BOOK]					#1122
#1122	#1050 [MEMO]					#1122
#1127	#1060 [DOCUMENT]		C:\WORD\PROJ.DOC		PROJECT PLAN	#1101
#1019	# 210 [FIELD]	EMPLOYED BY				
# 210	# 111 {TYPE]	COLUMN				
# 111	# 111 [TYPE]	TYPE				

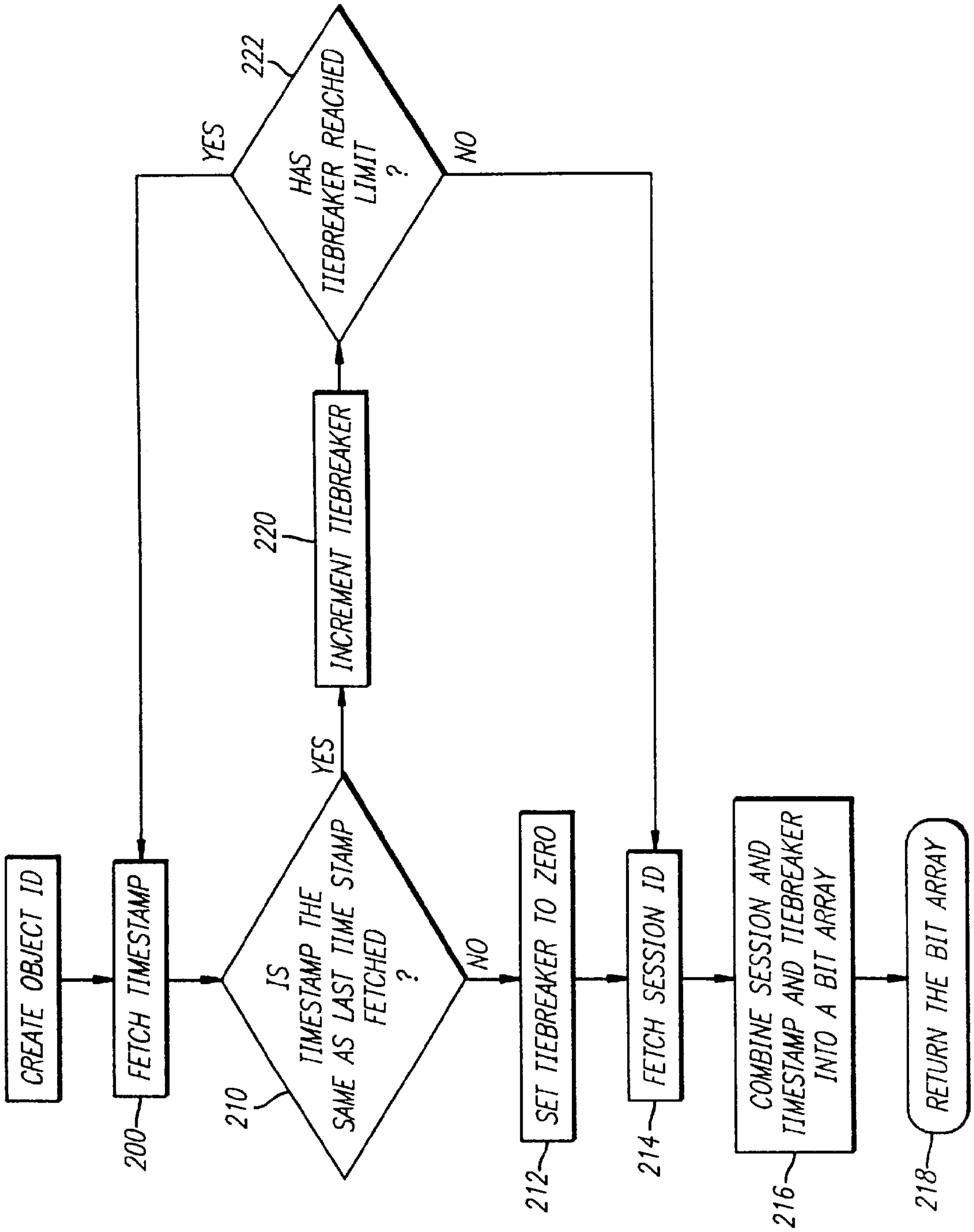
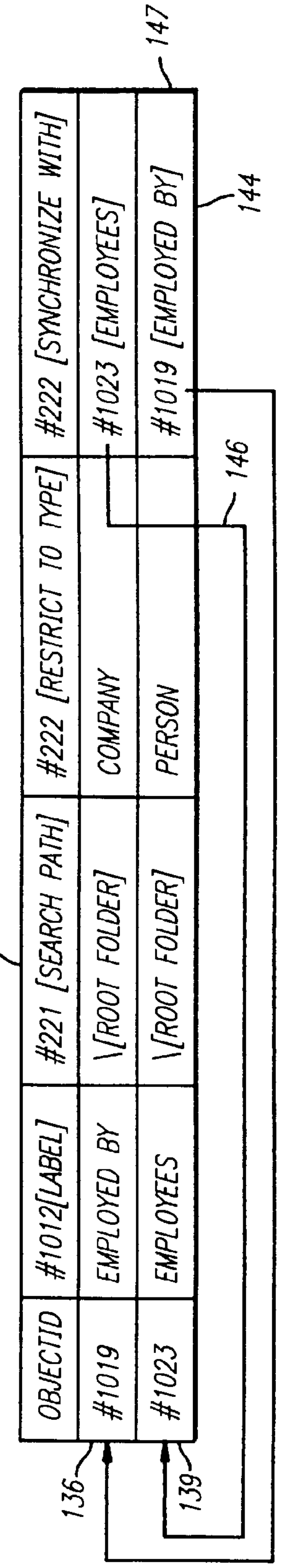
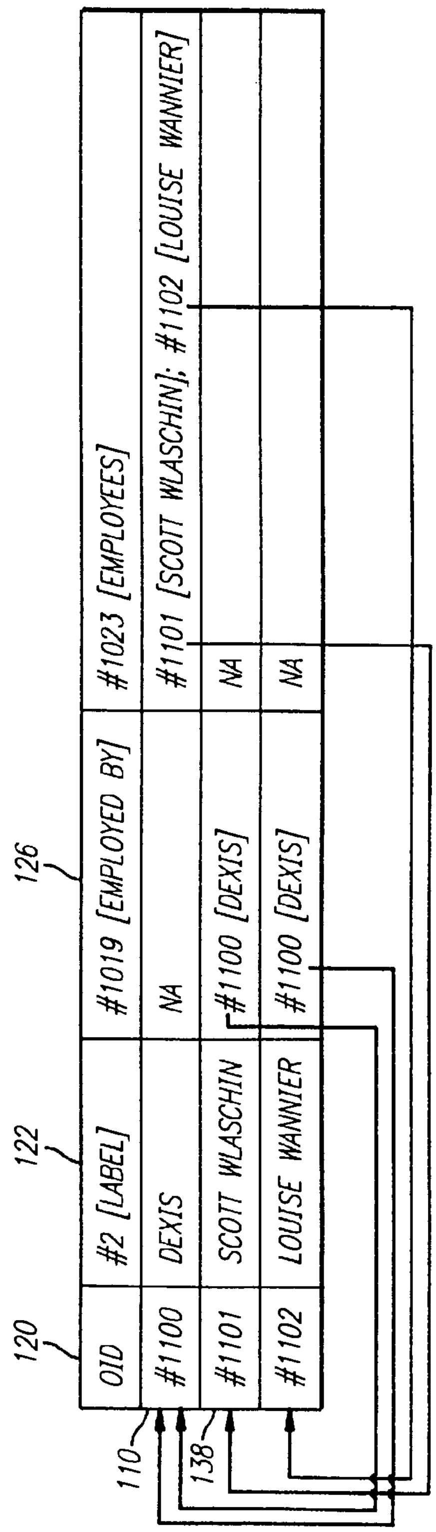

CREATE OBJECT ID
200
FETCH TIMESTAMP
210
IS TIMESTAMP THE SAME AS LAST TIME STAMP FETCHED ?
YES
NO
220
INCREMENT TIEBREAKER
222
HAS TIEBREAKER REACHED LIMIT ?
YES
NO
212
SET TIEBREAKER TO ZERO
214
FETCH SESSION ID
216
COMBINE SESSION AND TIMESTAMP AND TIEBREAKER INTO A BIT ARRAY
218
RETURN THE BIT ARRAY

FIG. 4



OBJECTID
#1012[LABEL]
#221 [SEARCH PATH]
#222 [RESTRICT TO TYPE]
#222 [SYNCHRONIZE WITH]
#1019
EMPLOYED BY
\[ROOT FOLDER]
COMPANY
#1023 [EMPLOYEES]
#1023
EMPLOYEES
\[ROOT FOLDER]
PERSON
#1019 [EMPLOYED BY]
136
139
144
146
146
147

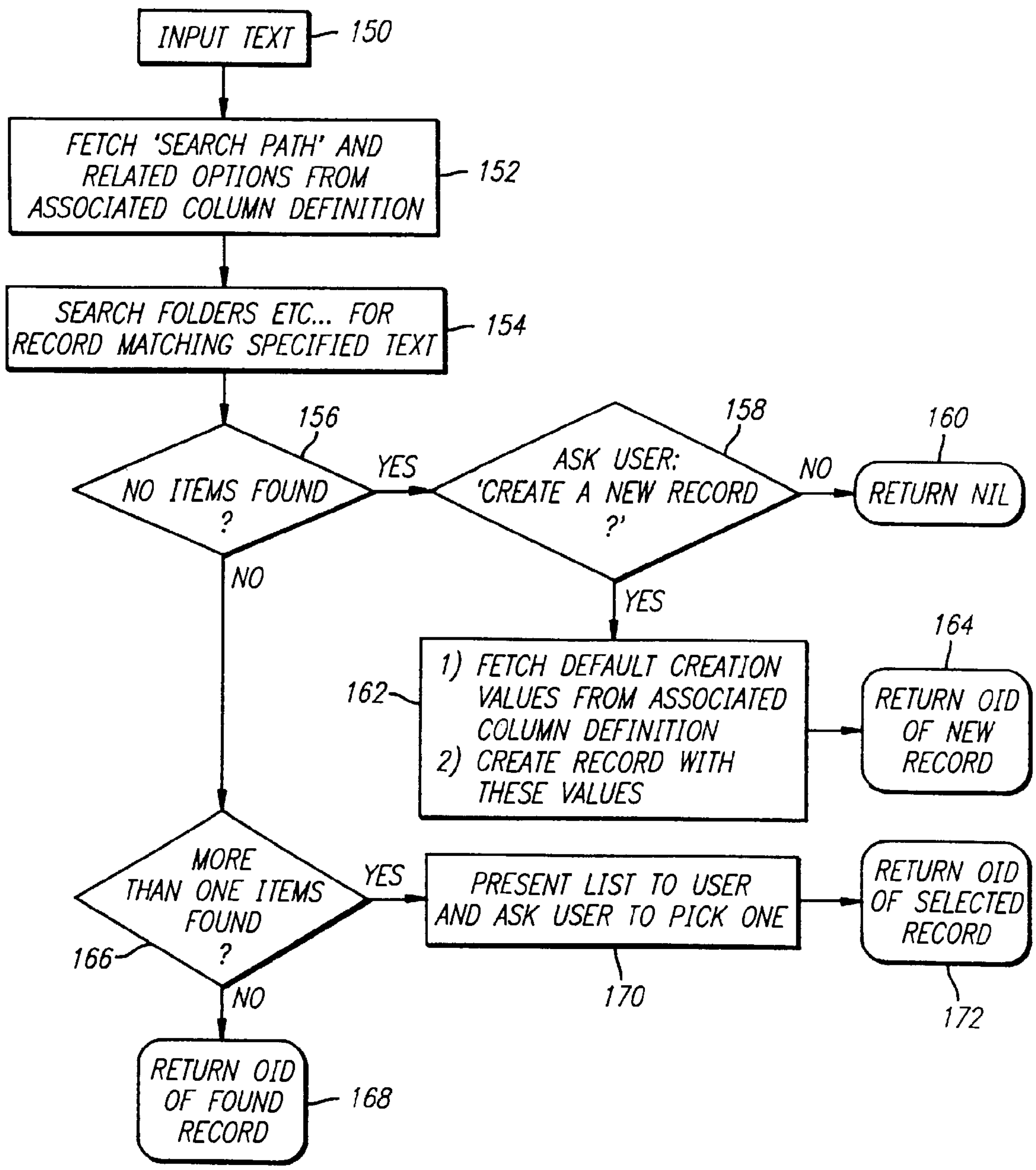
FIG. 5



OID
#2 [LABEL]
#1019 [EMPLOYED BY]
#1023 [EMPLOYEES]
#1100
DEXIS
NA
#1101 [SCOTT WLASCHIN]; #1102 [LOUISE WANNIER]
#1101
SCOTT WLASCHIN
#1100 [DEXIS]
NA
#1102
LOUISE WANNIER
#1100 [DEXIS]
NA
110
120
122
126
138

FIG. 7b

FIG. 6


INPUT TEXT
150
FETCH 'SEARCH PATH' AND RELATED OPTIONS FROM ASSOCIATED COLUMN DEFINITION
152
SEARCH FOLDERS ETC... FOR RECORD MATCHING SPECIFIED TEXT
154
156
NO ITEMS FOUND ?
YES
NO
158
ASK USER: 'CREATE A NEW RECORD ?'
NO
160
RETURN NIL
YES
164
162
1) FETCH DEFAULT CREATION VALUES FROM ASSOCIATED COLUMN DEFINITION
2) CREATE RECORD WITH THESE VALUES
RETURN OID OF NEW RECORD
MORE THAN ONE ITEMS FOUND ?
YES
166
PRESENT LIST TO USER AND ASK USER TO PICK ONE
RETURN OID OF SELECTED RECORD
NO
170
172
RETURN OID OF FOUND RECORD
168

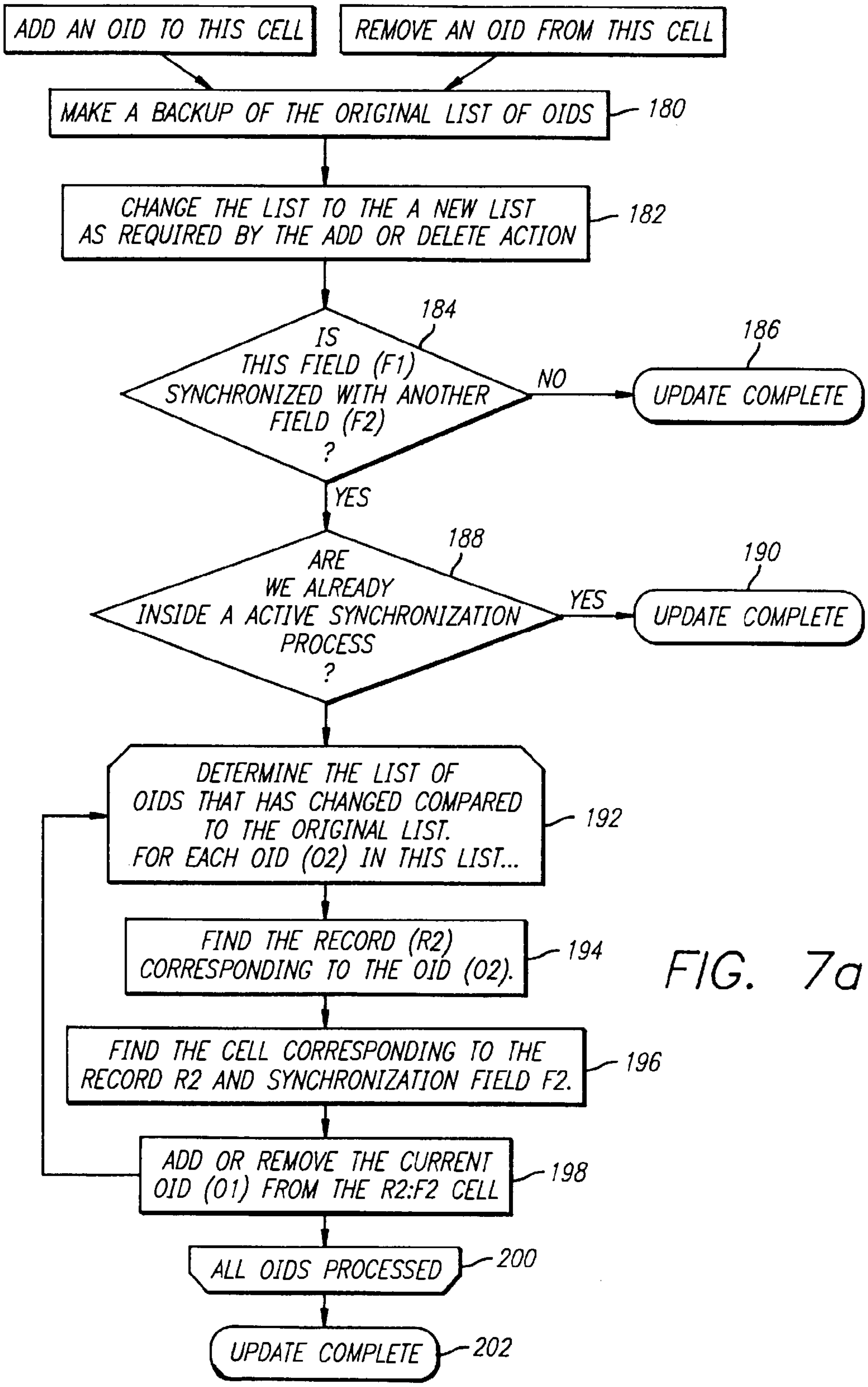

ADD AN OID TO THIS CELL
REMOVE AN OID FROM THIS CELL
MAKE A BACKUP OF THE ORIGINAL LIST OF OIDS
180
CHANGE THE LIST TO THE A NEW LIST AS REQUIRED BY THE ADD OR DELETE ACTION
182
IS THIS FIELD (F1) SYNCHRONIZED WITH ANOTHER FIELD (F2) ?
184
NO
UPDATE COMPLETE
186
YES
ARE WE ALREADY INSIDE A ACTIVE SYNCHRONIZATION PROCESS ?
188
YES
UPDATE COMPLETE
190
DETERMINE THE LIST OF OIDS THAT HAS CHANGED COMPARED TO THE ORIGINAL LIST. FOR EACH OID (O2) IN THIS LIST...
192
FIND THE RECORD (R2) CORRESPONDING TO THE OID (O2).
194
FIND THE CELL CORRESPONDING TO THE RECORD R2 AND SYNCHRONIZATION FIELD F2.
196
ADD OR REMOVE THE CURRENT OID (O1) FROM THE R2:F2 CELL
198
ALL OIDS PROCESSED
200
UPDATE COMPLETE
202

FIG. 7a

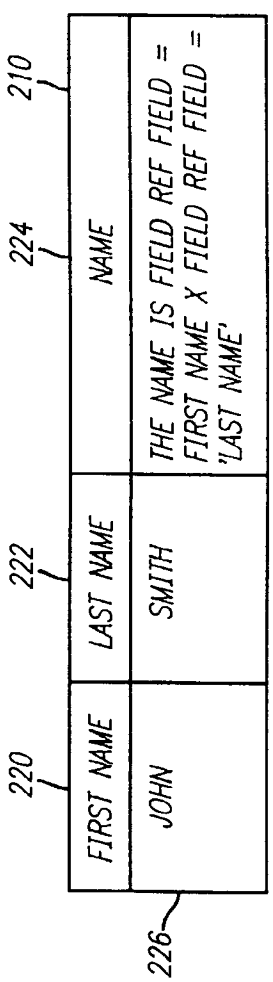


FIG. 8a

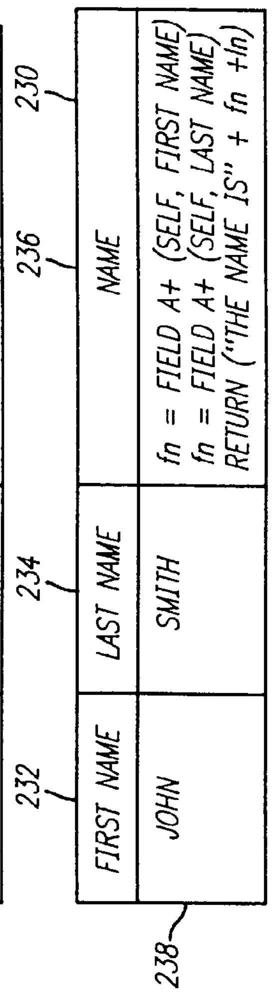


FIG. 8b

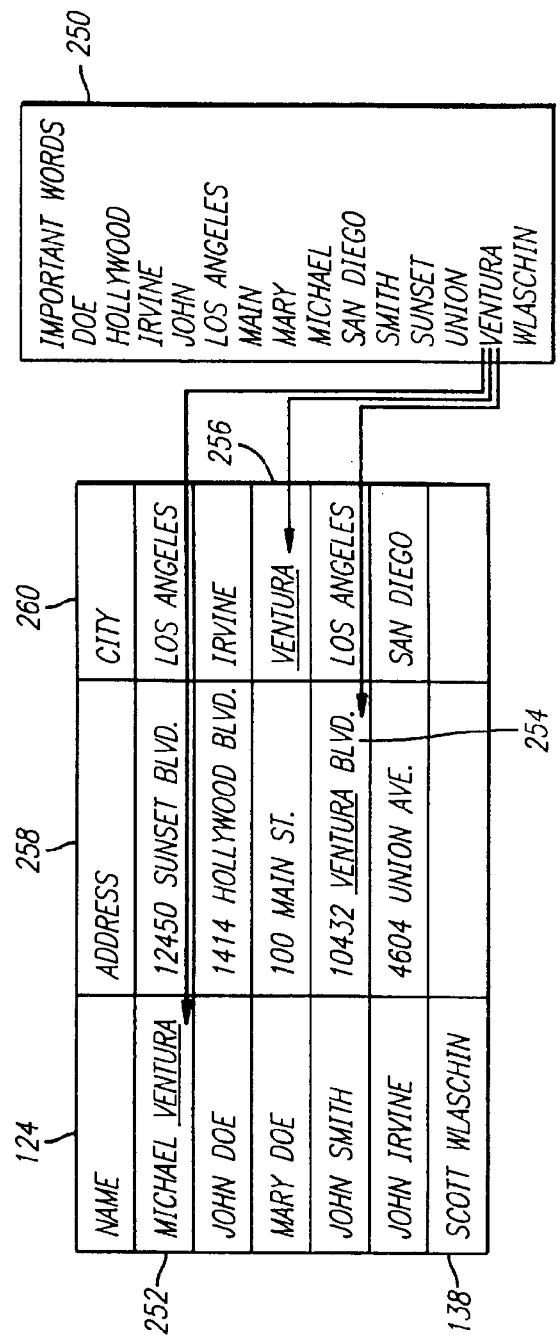
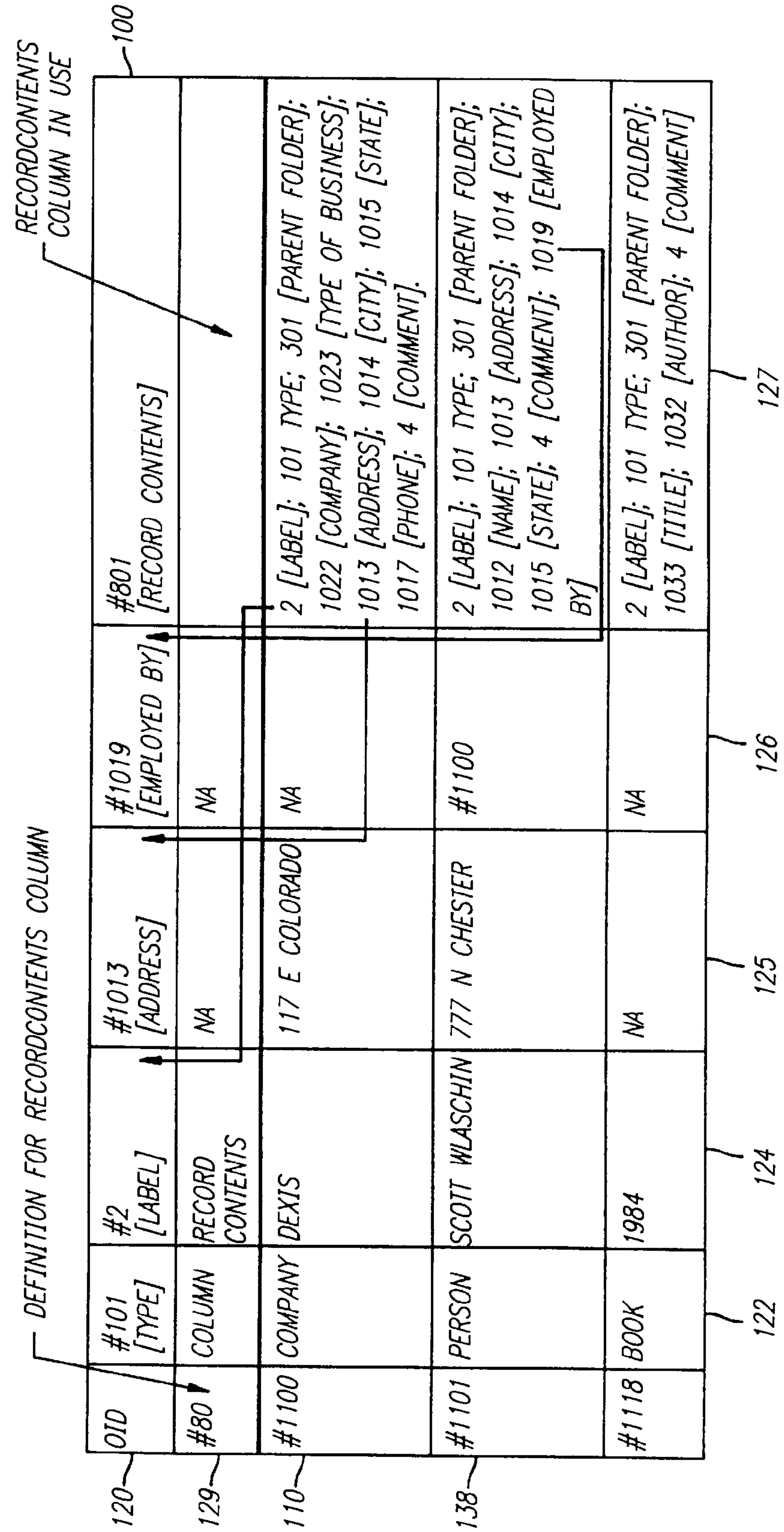


FIG. 11

FIG. 9



DEFINITION FOR RECORDCONTENTS COLUMN
RECORDCONTENTS COLUMN IN USE
100
OID
#101 [TYPE]
#2 [LABEL]
#1013 [ADDRESS]
#1019 [EMPLOYED BY]
#801 [RECORD CONTENTS]
#80
COLUMN
RECORD CONTENTS
NA
NA
#1100
COMPANY
DEXIS
117 E COLORADO
NA
2 [LABEL]; 101 TYPE; 301 [PARENT FOLDER]; 1022 [COMPANY]; 1023 [TYPE OF BUSINESS]; 1013 [ADDRESS]; 1014 [CITY]; 1015 [STATE]; 1017 [PHONE]; 4 [COMMENT].
#1101
PERSON
SCOTT WLASCHIN
777 N CHESTER
#1100
2 [LABEL]; 101 TYPE; 301 [PARENT FOLDER]; 1012 [NAME]; 1013 [ADDRESS]; 1014 [CITY]; 1015 [STATE]; 4 [COMMENT]; 1019 [EMPLOYED BY]
#1118
BOOK
1984
NA
NA
2 [LABEL]; 101 TYPE; 301 [PARENT FOLDER]; 1033 [TITLE]; 1032 [AUTHOR]; 4 [COMMENT]
120
129
110
138
122
124
125
126
127

FIG. 10

DEFINITIONS FOR FOLDER RELATED COLUMNS

OBJECTID	#101 [TYPE]	#2 [LABEL]	#301 [PARENT FOLDER]	#320 [FOLDERCHILDREN]
#301	FIELD	PARENT FOLDER	NA	NA
#320	FIELD	FOLDERCHILDREN	NA	NA
#1100	COMPANY	DEXIS	#1070 [CONTACTS]	NA
#1101	PERSON	SCOTT WLASCHIN	#1070 [CONTACTS]	NA
#1070	FOLDER	CONTACTS	NA	1100 [DEXIS]; 1101 [SCOTT WLASCHIN]; 1102 [LOUISE WANNIER]; ETC.

138

244

240

242

FOLDERCHILDREN IN USE AS AN ATTRIBUTE OF A FOLDER OBJECT

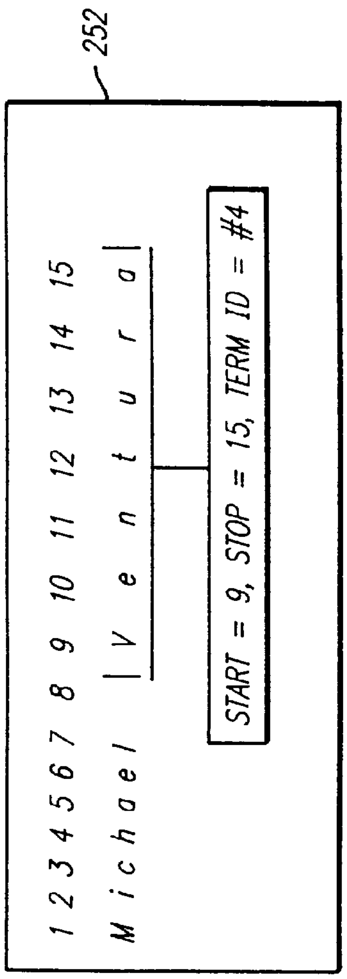


FIG. 12

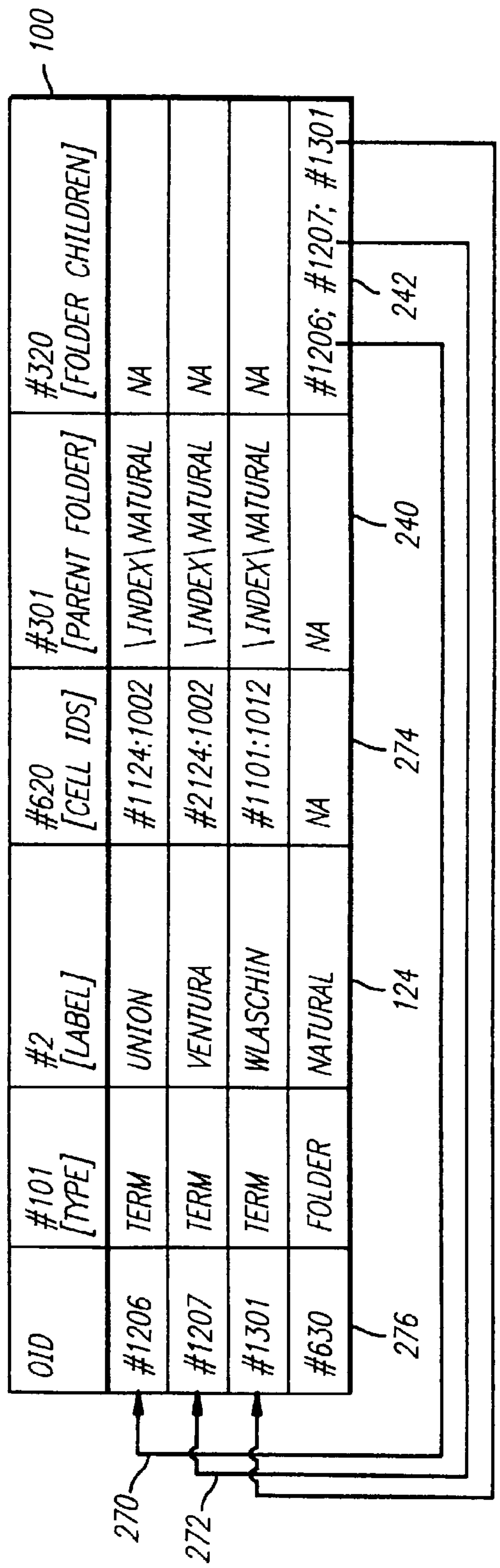


FIG. 13

Microsoft Corp. Exhibit 1001 Page 12

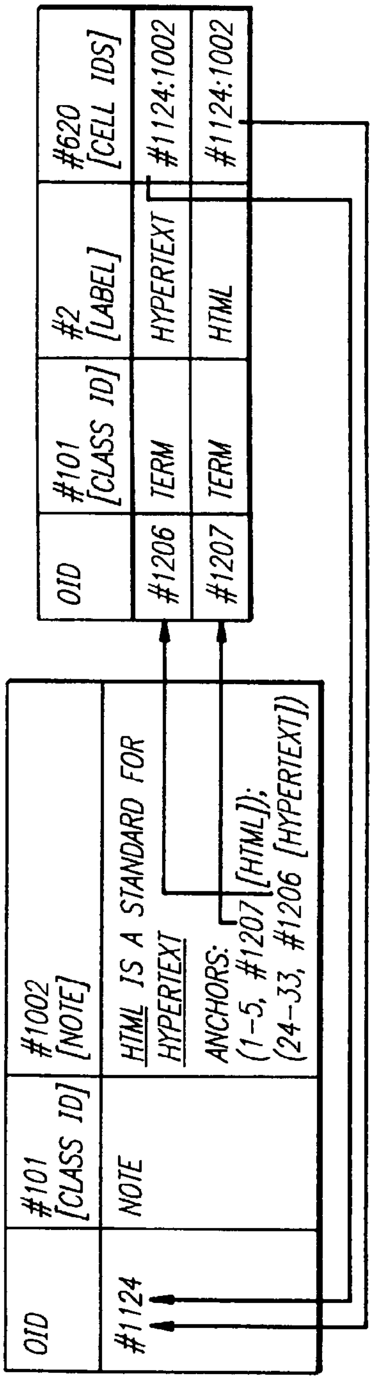

OID
#101 [CLASS ID]
#2 [LABEL]
#620 [CELL IDS]
#1206
TERM
HYPERTEXT
#1124:1002
#1207
TERM
HTML
#1124:1002
OID
#101 [CLASS ID]
#1002 [NOTE]
#1124
NOTE
HTML IS A STANDARD FOR HYPERTEXT
ANCHORS:
(1-5, #1207 [HTML]);
(24-33, #1206 [HYPERTEXT])

FIG. 14

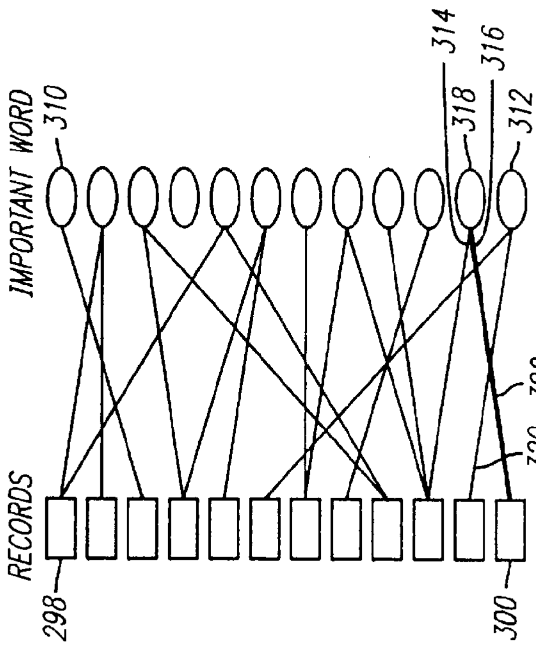

IMPORTANT WORD
310
314
316
318
312
RECORDS
298
300
320
322

FIG. 15

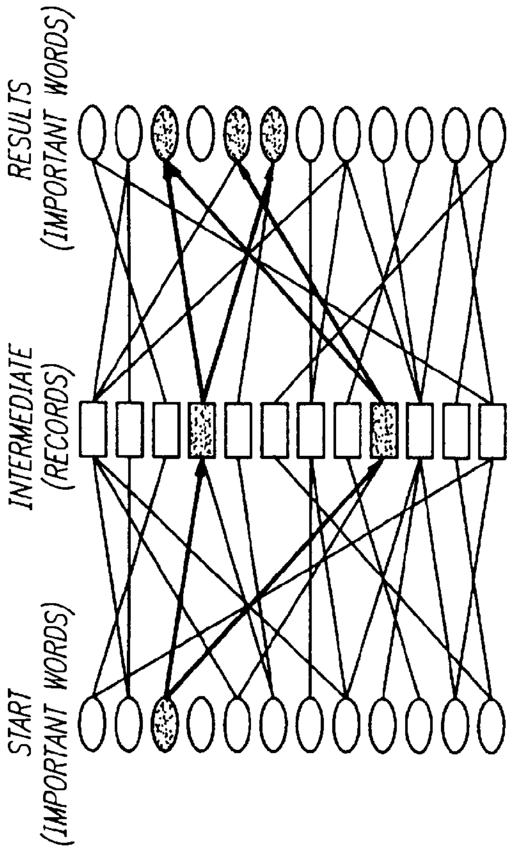


FIG. 16a

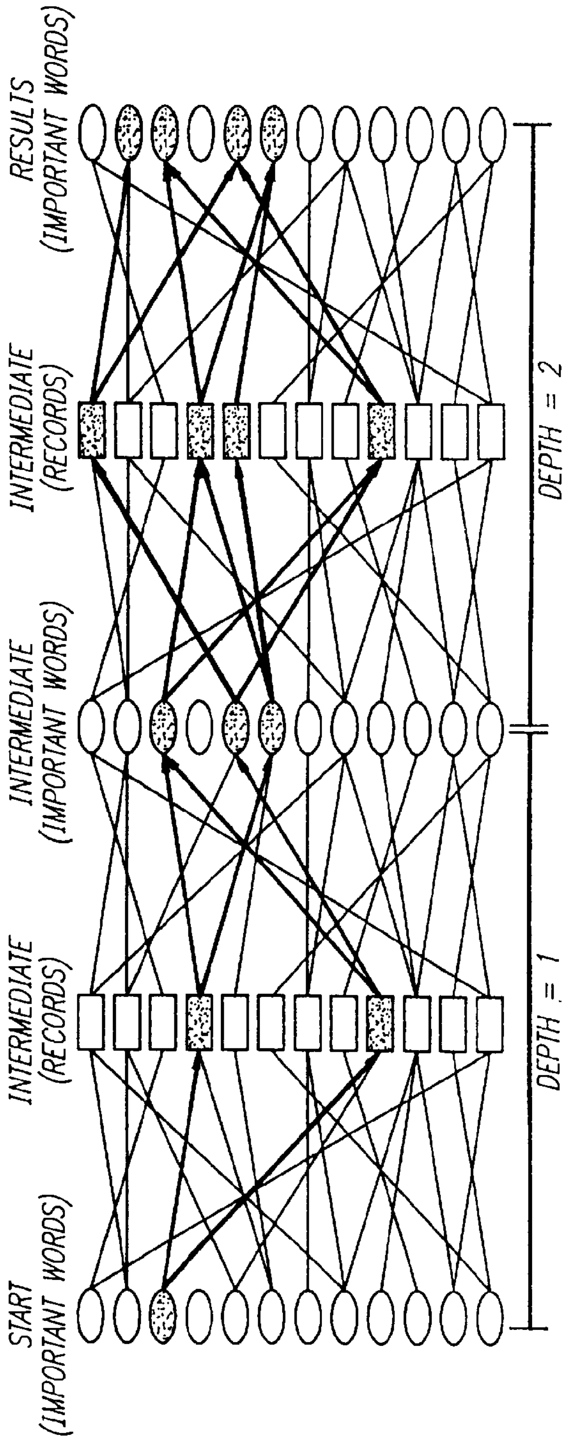


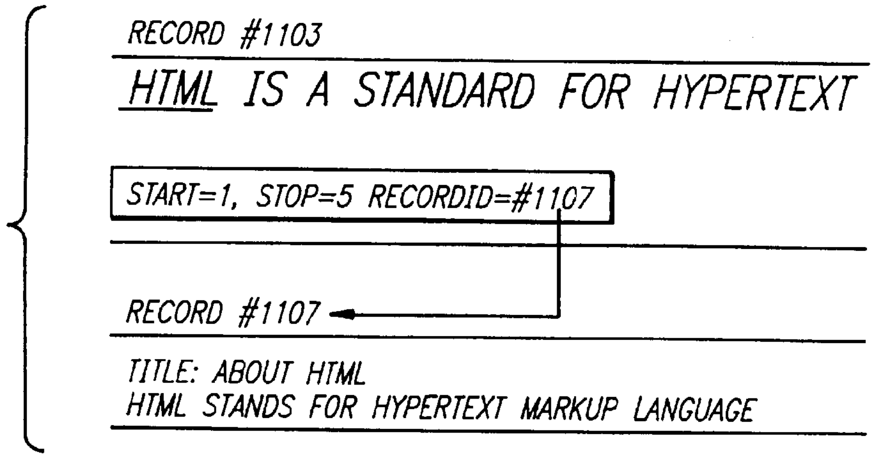
FIG. 16b

FIG. 17

OID	#101 [CLASS ID]	#2 [LABEL]	#621 [PARENT CONCEPT]	#701 [CONCEPT NAME]	#702 [SYNONYMS]	#703 [MORE SPECIFIC TERMS]	#704 [MORE GENERAL TERMS]	#705 [SEE ALSO]
1203	TERM	IBM	#1300 [IBM]					
1204	TERM	INTERNATIONAL BUSINESS MACHINES	#1300 [IBM]					
1300	CONCEPT	IBM		#1203 [IBM]	#1203 [IBM]: #1204 [INTERNATIONAL BUSINESS MACHINES]	#1301 [IBM PC] WEIGHT=100%	#1303 [COMPUTER COMPANIES], WEIGHT=60%	#1302 [MICROSOFT], WEIGHT=70%
1301	CONCEPT	IBM PC			#1300 [IBM], WEIGHT=50%			
1302	CONCEPT	MICROSOFT						#1300 [IBM], WEIGHT=70%
1303	CONCEPT	COMPUTER COMPANIES					#1300 [IBM], WEIGHT=100%	

100 372 122 124 352 354 356 358 360 362 364 366 350 368 370

FIG. 18 PRIOR ART



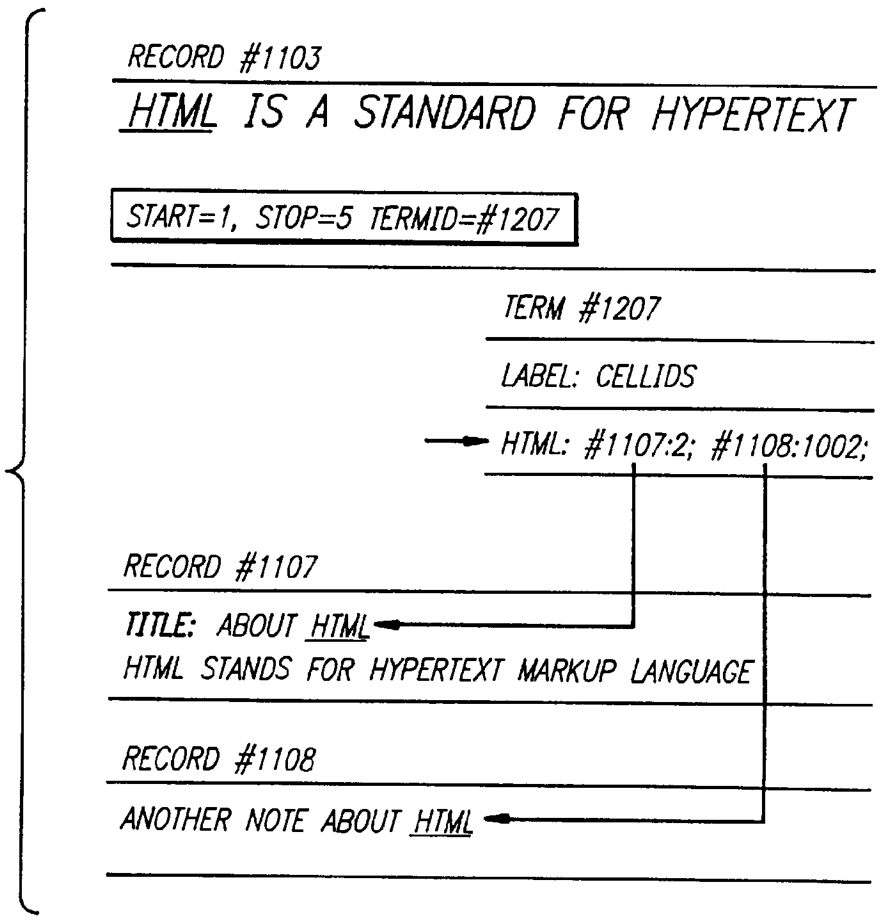


FIG. 19

FIG. 20a PRIOR ART

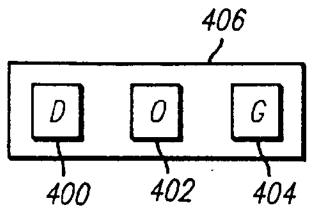


FIG. 20b PRIOR ART

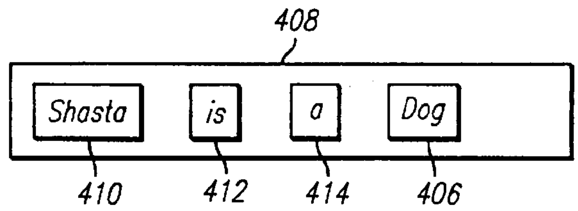


FIG. 20c PRIOR ART

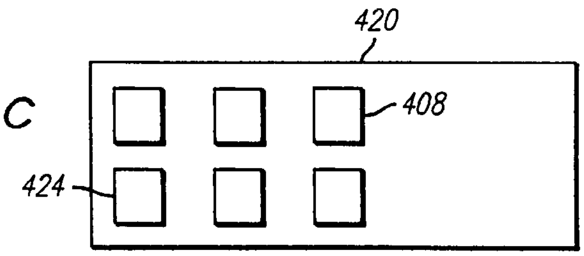


FIG. 21

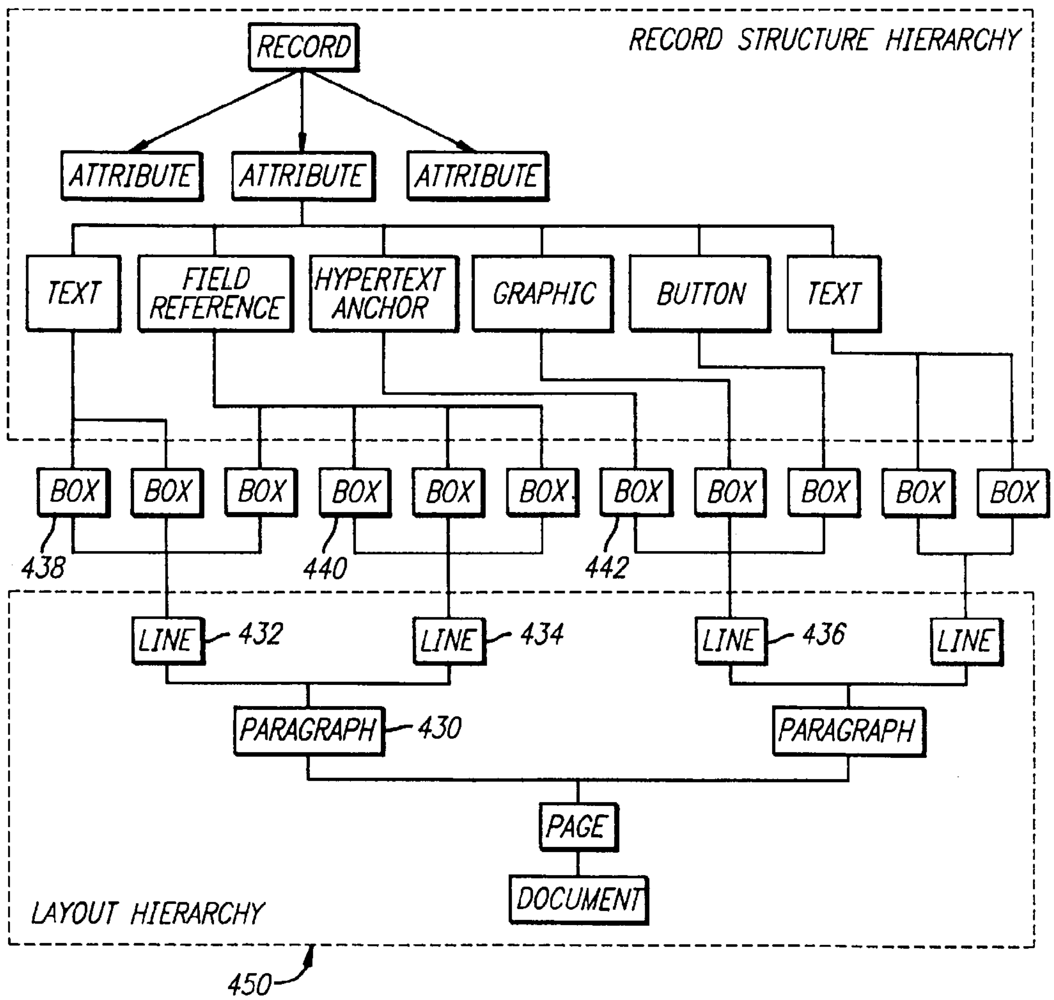


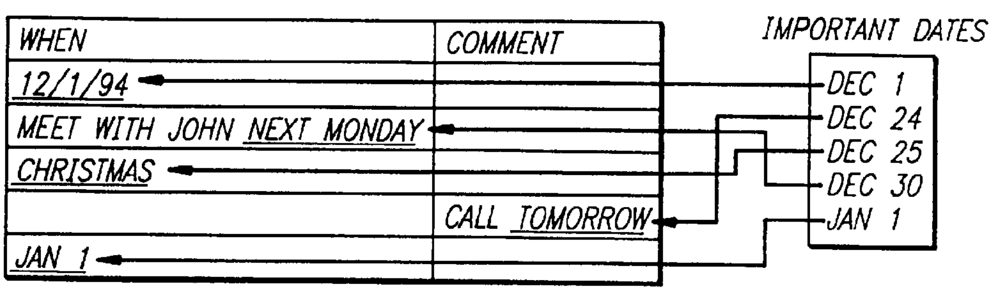
FIG. 22a
PRIOR ART

KEY PHRASE	SORTED UNDER
100	'1'
1984	'1', THEN '9'
20	'2', THEN '0'
3	'3'
JOHN SMITH	'J'
THE BIG OAK	'T'

FIG. 22b

KEY PHRASE	SORTED UNDER
3	3
20	20
100	100
1984	1984
THE BIG OAK	'B'-BIG
JOHN SMITH	'J'-JOHN
1984	'N'-NINETEEN EIGHTY FOUR
THE BIG OAK	'O'-OAK
100	'O'-ONE HUNDRED
1984	'O'-ONE THOUSAND NINE HUNDRED...
JOHN SMITH	'S'-SMITH
3	'T'-THREE
20	'T'-TWENTY
2	'T'-TWO

FIG. 23



METHOD AND APPARATUS FOR IMPROVED INFORMATION STORAGE AND RETRIEVAL SYSTEM

This is a continuation of application Ser. No. 08/383,752 filed Mar. 28, 1995 and now U.S. Pat. No. 5,729,730.

RELATED APPLICATIONS

The present application is related to the application entitled "Method and Apparatus for a Physical Storage Architecture for a Shared File Environment," filed Feb. 3, 1995, Ser. No. 08/383,752 now U.S. Pat. No. 5,729,730, which is herein incorporated by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to a method and apparatus for storing, retrieving, and distributing various kinds of data, and more particularly, to an improved database architecture and method for using the same.

2. Art Background

Over the past 30 years, computers have become increasingly important in storing and managing information. During this time, many database products have been developed to allow users to store and manipulate information and to search for desired information. The continuing growth of the information industry creates a demand for more powerful databases.

The database products have evolved over time. Initially, databases comprised a simple "flat file" with an associated index. Application programs, as opposed to the database program itself, managed the relationships between these files and a user typically performed queries entirely at the application program level. The introduction of relational database systems shifted many tasks from applications programs to database programs. The currently existing database management systems comprise two main types, those that follow the relational model and those that follow the object oriented model.

The relational model sets out a number of rules and guidelines for organizing data items, such as data normalization. A relational database management system (RDBMS) is a system that adheres to these rules. RDBMS databases require that each data item be uniquely classified as a particular instance of a 'relation'. Each set of relations is stored in a distinct 'table'. Each row in the table represents a particular data item, and each column represents an attribute that is shared over all data items in that table.

The pure relational model places number of restrictions on data items. For example, each data item cannot have attributes other than those columns described for the table. Further, an item cannot point directly to another item. Instead, 'primary keys' (unique identifiers) must be used to reference other items. Typically, these restrictions cause RDBMS databases to include a large number of tables that require a relatively large amount of time to search. Further, the number of tables occupies a large amount of computer memory.

The object oriented database model, derived from the object-oriented programming model, is an alternative to the relational model. Like the relational model, each data item must be classified uniquely as belonging to a single class, which defines its attributes. Key features of the object-oriented model are: 1) each item has a unique system-generated object identification number that can be used for exact retrieval; 2) different types of data items can be stored together; and 3) predefined functions or behavior can be created and stored with a data item.

Apart from the limitations previously described, both the relational and object oriented models share important limitations with regard to data structures and searching. Both models require data to be input according to a defined field structure and thus do not completely support full text data entry. Although some databases allow records to include a text field, such text fields are not easily searched. The structural requirements of current databases require a programmer to predefine a structure and subsequent date entry must conform to that structure. This is inefficient where it is difficult to determine the structure of the data that will be entered into a database.

Conversely, word and image processors that allow unstructured data entry do not provide efficient data retrieval mechanisms and a separate text retrieval or data management tool is required to retrieve data. Thus, the current information management systems do not provide the capability of integrating full text or graphics data entry with the searching mechanisms of a database.

The present invention overcomes the limitations of both the relational database model and object oriented database model by providing a database with increased flexibility, faster search times and smaller memory requirements and that supports text attributes. Further, the database of the present invention does not require a programmer to preconfigure a structure to which a user must adapt data entry. Many algorithms and techniques are required by applications that deal with these kinds of information. The present invention provides for the integration, into a single database engine, of support for these techniques, and shifts the programming from the application to the database, as will be described below. The present invention also provides for the integration, into a single database, of preexisting source files developed under various types of application programs such as other databases, spreadsheets and word processing programs. In addition, the present invention allows users to control all of the data that are relevant to them without sacrificing the security needs of a centralized data repository.

SUMMARY OF THE INVENTION

The present invention improves upon prior art information search and retrieval systems by employing a flexible, self-referential table to store data. The table of the present invention may store any type of data, both structured and unstructured, and provides an interface to other application programs such as word processors that allows for integration of all the data for such application programs into a single database. The present invention also supports a variety of other features including hypertext.

The table of the present invention comprises a plurality of rows and columns. Each row has an object identification number (OID) and each column also has an OID. A row corresponds to a record and a column corresponds to an attribute such that the intersection of a row and a column comprises a cell that may contain data for a particular record related to a particular attribute. A cell may also point to another record. To enhance searching and to provide for synchronization between columns, columns are entered as rows in the table and the record corresponding to a column contains various information about the column. This renders the table self referential and provides numerous advantages, as will be discussed in this Specification.

The present invention includes an index structure to allow for rapid searches. Text from each cell is stored in a key

word index which itself is stored in the table. The text cells include pointers to the entries in the key word index and the key word index contains pointers to the cells. This two way association provides for extended queries. The invention further includes weights and filters for such extended queries.

The present invention includes a thesaurus and knowledge base that enhances indexed searches. The thesaurus is stored in the table and allows a user to search for synonyms and concepts and also provides a weighting mechanism to rank the relevance of retrieved records.

An application support layer includes a word processor, a password system, hypertext and other functions. The novel word processor of the present invention is integrated with the table of the present invention to allow cells to be edited with the word processor. In addition, the table may be interfaced with external documents which allows a user to retrieve data from external documents according to the enhanced retrieval system of the present invention.

These and numerous other advantages of the present invention will be apparent from the following description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a functional block diagram illustrating one possible computer system incorporating the teachings of the present invention.

FIG. **2** is a block diagram illustrating the main components of the present invention.

FIG. **3** illustrates the table structure of the database of the present invention.

FIG. **4** is a flow chart for a method of computing object identification numbers (OID's) that define rows and columns in the table of FIG. **1**.

FIG. **5** is a part of the table of FIG. **2** illustrating the column synchronization feature of the present invention.

FIG. **6** is a flow chart for a method of searching the table of FIG. **2**.

FIG. **7***a* is a flow chart for synchronizing columns of the table of FIG. **2**.

FIG. **7***b* illustrates the resnlts of column synchronization.

FIG. **8***a* illustrates a reference within one column to another column.

FIG. **8***b* illustrates an alternate embodiment for referring to another column within a column.

FIG. **9** illustrates a "Record Contents" column of the present invention that indicates which columns of a particular record have values.

FIG. **10** illustrates a folder structure that organizes records. The folder structure is stored within the table of FIG. **2**.

FIG. **11** illustrates the correspondence between cells of the table of FIG. **2** and a sorted key word index.

FIG. **12** illustrate the "anchors" within a cell that relate a word in a cell to a key word index record.

FIG. **13** illustrates key word index records stored in the table of FIG. **2**.

FIG. **14** illustrates the relationship between certain data records and key word index records.

FIG. **15** illustrates the relationship of FIG. **14** in graphical form.

FIG. **16***a* illustrates an extended search in graphical form.

FIG. **16***b* illustrates a further extended search in graphical form.

FIG. **17** illustrates the thesaurus structure of the present invention stored in the table of FIG. **2**.

FIG. **18** illustrates prior art hypertext.

FIG. **19** illustrates the hypertext features of the present invention.

FIG. **20***a* illustrates a character and word box structure of the word processor of the present invention.

FIG. **20***b* illustrates the word and horizontal line box structure of the word processor of the present invention.

FIG. **20***c* illustrates the vertical box structure of the word processor of the present invention.

FIG. **21** illustrates the box tree structure of the word processor of the present invention.

FIG. **22***a* illustrates the results of a prior art sorting algorithm.

FIG. **22***b* illustrates the results of a sorting alogrithm according to the present invention.

FIG. **23** illustrates the correspondence between cells of the table of FIG. **2** and a sorted date index.

NOTATION AND NOMENCLATURE

The detailed descriptions which follow are presented largely in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means nsed by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein which form part of the present invention; the operations are machine operations. Useful machines for performing the operations of the present invention include general purpose digital computers or other similar digital devices. In all cases there should be borne in mind the distinction between the method operations in operating a computer and the method of computation itself. The present invention relates to method steps for operating a computer in processing electrical or other (e.g., mechanical, chemical) physical signals to generate other desired physical signals.

The present invention also relates to apparatus for performing these operations. This apparatus may be specially constructed for the required purposes or it may comprise a general purpose computer as selectively activated or reconfigured by a computer program stored in the computer. The algorithms presented herein are not inherently related to a particular computer or other apparatus. In particular, various general purpose machines may be used with programs written in accordance with the teachings herein, or it may

prove more convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description given below.

DETAILED DESCRIPTION OF THE INVENTION

The present invention discloses methods and apparatus for data storage, manipulation and retrieval. Although the present invention is described with reference to specific block diagrams, and table entries, etc., it will be appreciated by one of ordinary skill in the art that such details are disclosed simply to provide a more thorough understanding of the present invention. It will therefore be apparent to one skilled in the art that the present invention may be practiced without these specific details.

System Hardware

Referring to FIG. **1**, the hardware configuration of the present invention is conceptually illustrated. FIG. **1** illustrates an information storage and retrieval system structnred in accordance with the teachings of the present invention. As illustrated, the information storage and retrieval system includes a computer **23** which comprises four major components. The first of these is an input/output (I/O) circuit **22**, which is used to communicate information in appropriately structured form to and from other portions of the computer **23**. In addition, computer **20** includes a central processing unit (CPU) **24** coupled to the I/O circuit **22** and to a memory **26**. These elements are those typically found in most computers and, in fact, computer **23** is intended to be representative of a broad category of data processing devices.

Also shown in FIG. **1** is a keyboard **30** for inputting data and commands into computer **23** through the I/O circuit **22**, as is well known. Similarly, a CD ROM **34** is coupled to the I/O circuit **22** for providing additional programming capacity to the system illustrated in FIG. **1**. It will be appreciated that additional devices may be coupled to the computer **20** for storing data, such as magnetic tape drives, buffer memory devices, and the like. A device control **36** is coupled to both the memory **26** and the I/O circuit **22**, to permit the computer **23** to communicate with multi-media system resources. The device control **36** controls operation of the multi-media resources to interface the multi-media resources to the computer **23**.

A display monitor **43** is coupled to the computer **20** through the I/O circuit **22**. A cursor control device **45** includes switches **47** and **49** for signally the CPU **24** in accordance with the teachings of the present invention. A cursor control device **45** (commonly referred to a "mouse") permits a user to select various command modes, modify graphic data, and input other data utilizing switches **47** and **49**. More particularly, the cursor control device **45** permits a user to selectively position a cursor **39** at any desired location on a display screen **37** of the display **43**. It will be appreciated that the cursor control device **45** and the keyboard **30** are examples of a variety of input devices which may be utilized in accordance with the teachings of the present invention. Other input devices, including for example, trackballs, touch screens, data gloves or other virtual reality devices may also be used in conjunction with the invention as disclosed herein.

System Architecture

FIG. **2** is a block diagram of the information storage and retrieval system of the present invention. As illustrated in the

Figure, the present invention includes an internal database **52** that further includes a record oriented database **74** and a free-text database **76**. The database **52** may receive data from a plurality of external sources **50**, including word processing documents **58**, spreadsheets **60** and database files **62**. As will be described more fully below, the present invention includes an application support system that interfaces the external sources **50** with the database **52**.

To efficiently retrieve information stored in the database **52**, a plurality of indexes **54** including a keyword index **78** and other types of indexes such as phonetic, special sorting for other languages, and market specific such as chemical, legal and medical, store sorted information provided by the database **52**. To organize the information in the indexes **54**, a knowledge system **56** links information existing in the indexes **54**.

The organization illnstrated in FIG. **2** is for conceptnal purposes and, in actuality, the database **52**, the indexes **54** and the knowledge system **56** are stored in the same table, as will be described more fully below. This Specification will first describe the structure and features of the database **52**. Next, the Specification will describe the index **54** and its implementation for searching the database **52**. The Specification will then describe the knowledge system **56** that further enhances the index **54** by providing synonyms and other elements. Finally, the Specification will describe an interface between the external application programs **50** and the database **52**, including a novel structured word processor and a novel password scheme.

FIG. **3** illustrates the storage and retrieval structure of the present invention. The storage and retrieval structure of the present invention comprises a table **100**. The structure of the table **100** is a logical structure and not necessarily a physical structure. Thus, the memories **26** and **32** configured according to the teachings of the present invention need not store the table **100** contiguously.

The table **100** further comprises a plurality of rows **110** and a plurality of columns **120**. A row corresponds to a record while a column corresponds to an attribute of a record and the defining characteristics of the column are stored in a row **108**. The intersection of a row and a column comprises a particular cell.

Each row is assigned a unique object identification number (OID) stored in column **120** and each column also is assigned a unique OID, indicated in brackets and stored in row **108**. For example, row **110** has an OID equal to **1100** while the column **122** has an OID equal to **101**. As will be described more fully below, the OID's for both rows and columns may be used as pointers and a cell **134** may store an OID. The method for assigning the OID's will also be discussed below.

As illustrated in FIG. **3**, each row, corresponding to a record, may include information in each column. However, a row need not, and generally will not, have data stored in every column. For example, row **110** corresponds to a company as shown in a cell **130**. Since companies do not have titles, cell **132** is unused.

The type of information associated with a column is known as a 'domain'. Standard domains supported in most database systems include text, number, date, and Boolean. The present invention includes other types of domains such as the OID domain that points to a row or column. The present invention further supports 'user-defined' domains, whereby all the behavior of the domain can be determined by a user or programmer. For example, a user may configure a domain to include writing to and reading from a storage medium and handling operations such as equality testing and comparisons.

According to the present invention, individual cells may be accessed according to their row and column OID's. Using the cell as the unit of storage improves many standard data management operations that previously required the entire object or record. Such operations include versioning, security, hierarchical storage management, appending to remote partitions, printing, and other operations.

Column Definitions

Each column has an associated column definition, which determines the properties of the column, such as the domain of the column, the name of the column, whether the column is required and other properties that may relate to a column. The table **100** supports columns that include unstructured, free text data.

The column definition is stored as a record in the table **100** of FIG. **3**. For example, the "Employed By" column **126** has a corresponding row **136**. The addition or rows that correspond to columns renders the table **100** self-referential. New columns may be easily appended to the table **100** by creating a new column definition record. The new column is then immediately available for use in existing records.

Dates

Dates can be specified numerically and textually. An example of a numerical date is "11/6/67" and an example of a textual date is "November 6, 1967." Textual entries are converted to dates using standard algorithms and lookup tables. A date value can store both original text and the associated date to which the text is converted, which allows the date value to be displayed in the format in which it was originally entered.

Numbers

Numeric values are classified as either a whole number (Integer) or fractional number. In the preferred embodiment, Integers are stored as variable length structures, which can represent arbitrarily large numbers. All data structures and indexes use this format which ensures that there are no limits in the system.

Fractional numbers are represented by a <numerator/denominator> pair of variable length integers. As with dates, a numeric value can store both the original text ("4½ inches") and the associated number (4.5). This allows the numeric value to be redisplayed in the format in which it was originally entered.

Type Definitions

A record can be associated with a 'record type'. The record type can be used simply as a category, but also can be used to determine the behavior of records. For example, the record type might specify certain columns that are required by all records of that type and, as with columns, the type definitions are stored as records in the table **100**. In FIG. **3**, column **122** includes the type definition for each record. The column **122** stores pointers to rows defining a particular column type. For example, the row **136** is a "Field" type column and contains a pointer in a cell **133** to a row **135** that defines "Field" type columns. The "Type Column" **122** of the row **135** points to a type called "Type," which is defined in a row **140**. "Type" has a type column that points to itself.

Record types, as defined by their corresponding rows, may constrain the values that a record of that type may contain. For example, the record type 'Person' may require that records of type 'Person' have a valid value in the 'Name' column, the 'Phone' column, and any other columns. The type of a record is an attribute of the record and thus may change at any time.

Creating a Unique OID

As previously described, the system must generate a unique OID when columns and rows are formed. FIG. **4** is a flow chart of the method for assigning OID's.

At block **200** of FIG. **4**, the CPU **24** running the database program stored in the memory **26** requests a timestamp from the operating system. At block **210**, the system determines whether the received timestamp is identical to a previous timestamp. If the timestamps are identical, block **210** branches to block **220** and a tiebreaker is incremented to resolve the conflict between the identical timestamps. At block **222**, the system determines whether the tiebreaker has reached its limit, and, if so, the system branches to block **200** to retrieve a new time stamp. Otherwise, the system branches to block **214** where the system requests a session identification which is unique to the user session.

In the preferred embodiment, the session identification is derived from the unique serial number of the application installed on the users machine. For certain OID's which are independent of any particular machine, the session identification may be used to determine the type of object. For example, dates are independent of any particular machine, and so an OID for a date may have a fixed session identification.

Returning to block **210**, if the timestamps are not identical, control passes to block **212** where the tiebreaker is set to zero and control then passes to block **214**. As previously described, at block **214**, the system requests a session identification which is unique to the user session. Control then passes to block **216** where the session identification, timestamp and tiebreaker are combined into a bit array, which becomes the OID. Since the OID is a variable length structure, any number of bits may be used, depending on the precision required, the resolution of the operating system clock, and the number of users. In the preferred embodiment, the OID is 64 bits long where the timestamp comprises the first 32 bits, the tiebreaker comprises the next 10 bits and the session identification comprises 22 bits.

The particular type of OID and its length is constant throughout a single database but may vary between databases. A flag indicating which type of OID to be used may be embedded in the header of each database.

OD Domains

OID domains are used to store OID's, which are pointers to other records. An efficient query can use these OID's to go directly to another record, rather than searching through columns.

If a user wishes to search a column to find a record or records with a certain item in the column, and does not know the OID of the item, the present invention includes a novel technique for determining an OID from the textual description. Conversion from text to an OID may also be necessary when a user is entering information into a record. For exmaple, in FIG. **3**, the user may be entering information in the "Employed By" column **126**, and wish to specify the text "DEXIS" and have it converted to OID #1100. For this purpose, special columns are required that provide a definition for how the search and conversion is performed.

FIG. **6** is a flow chart for searching the table **100** configured according to the structure illustrated in FIG. **5**. At block

150, a user enters text through the keyboard 30 or mouse 45 for a particular column that the user wishes to search. At block 152, the system retrieves the search path for the column to be searched from the information stored in column 146 as illustrated in FIG. 5. Continuing with the above example, a cell 146 in the row 136 contains the search path information for the "Employed By" column 126 of FIG. 3. The search path information for the "Employed By" field indicates that the folders called "\contacts" and "\departments" should be searched for a company with the dabel "DEXIS."

Returning to FIG. 5, the system searches the table 100 according to the retrieved search path information. For each folder specified in the search path, the routine searches for a record that has an entry in the label column 124 of FIG. 2 that is the same as the text being searched for, and is of the same class, as indicated in column 122 of FIG. 3. Folders will be further described below.

At block 156, the system determines whether it has found any items matching the user's search text. If no items have been found, at block 158, the system prompts the user on the display screen 37 to create a new record. If the user wishes to create a new record, control passes to block 162 and the system creates a new record. At block 164, the OID of the new record is returned. If the user does not wish to create a new record, a "NIL" string is returned, as shown at block 160.

If the system has located at least one item, the system determines whether it has found more than one item, as illustrated in block 166. If only one item has been located, its OID is returned at block 168. If more than one item has been located, the system displays the list of items to the user at block 170 and the user selects a record from the list. At block 172, the OID of the selected record is returned, which, in the above example, is #1100, the OID of the record for the company "DEXIS."

In alternate embodiments, various features may be added to the search mechanism as described with reference to FIG. 6. For example, further restrictions may be added to the search; the search may be related by allowing prefix matching or fuzzy matching instead of strict matching; and the search may be widened by using the 'associative search' techniques described below.

Two Way Synchronized Links

Records may have interrelationships and it is often desirable to maintain consistency between interrelated records. For example, a record including data for a company may include information regard employees of that company, as illustrated in row 110 of FIG. 3. Similarly, the employees that work for that company may have a record that indicates, by a pointer, their employer, as illustrated by row 138 of FIG. 3. Thus, the employee column of a company should point to employees whose employer column points to that company. The present invention includes a synchronization technique to ensure that whenever interrelated records are added or removed, the interrelationships between the columns are properly updated.

The system synchronizes interrelated records by adding a "Syuchronize With" column 144 to the table 100 as illustrated in FIG. 5. Since the value in the columns defines the relatedness between records, the rows 136 and 139 corresponding to columns contain information within the "Synchronize With" column 144 that indicates which other columns are to be synchronized with the columns corresponding to rows 136 and 139. With reference to FIG. 5, the

"Employed By" column 126 is synchronized with the "Employees" column by an OID pointer in the "Synchronize With" column 144 to the "Employees" column, represented by row 139. Similarly, the "Employees" column is synchronized with the "Employed By" column 136 by a pointer in the "Synchronize With" column 144 to the "Employed by" column 134, represented by row 136. Thus, whenever an employee changes companies, such that the employee's "Employed By" column changes, the "Employee" column of the previous employer is updated to eliminate the pointer to the ex-employee and, correspondingly, the addition of the employee in the "Employed By" field of the new employer. Synchronization may need to occur whenever a column is changed, whether by addition or subtraction of a reference to another column, or when entire records are added or eliminated from the table 100.

FIG. 7*a* is a flow chart for synchronizing records when a user adds or deletes a record. At block 180, the system makes a backup of the original list of references to other rows, which are simply the OID's of those other rows, so that it can later determine which OIDS have been added or removed. Only these changes need to be synchronized. At block 182, the system generates a new list of references by adding or deleting the specified OID. At block 184, the system determines whether the relevant column is synchronized with another column. If it is not, then the system branches to block 186 and the update is complete. If the column is synchronized with another column, the system determines whether it is already in a synchronization routine. If this were not done, the routine would get into an endless recursive loop. If the system is already in a synchronization routine, the system branches to 190 and the update is complete.

Otherwise, the system performs actual synchronization. At block 192, the system finds an OID that has been added or subtracted from the column (C1) of the record (R1) being altered. The system retrieves the record (R2) corresponding to the added or subtracted OID at block 194. The system determines the synchronization column (C2) of the column (C1) at block 196 and locates that field in the added or subtracted OID. For example, if an employer is fired from a job, and the employer's "Employed By" field changed accordingly, the system would look up the value of the "Synchronize With" column 144 for the "Employees" column which is contained in the cell 147 as illustrated in FIG. 5. Since cell 147 points to the "Employed By" field, the system locates the "Employed By" field of the record for the fired employee. At block 198 of FIG. 7*a*, the located cell, (R2:C2), is updated by adding or subtracting the OID. Continuing with the above example, the "Employed By" field of the employee would be changed to no longer point to the previous employer by simply removing the employer's OID from that field. The system branches back to block 192 to update any other OID additions or subtractions. If the system has processed all of the OID's, then the routine exits as illustrated at blocks 200 and 202.

FIG. 7*b* illustrates the results of column synchronization of the "Employed By" field and the "Employees" field. As shown, the pointers in the records of these two columns are consistent with one another.

Columns Within Columns

A column may contain within it a reference to another column in the same record. For example, a 'name' column may contain a reference to both a 'first name' and a 'last name' column. The value of the 'name' column can then be

reconstructed from the values of the other two columns. FIGS. **8***a* and **8***b* illustrate two possible implementations for reconstructing a value from one or more columns within the same record.

FIG. **8***a* illustrates a table **210** that includes a "First Name" column **220**, a "Last Name" column **222** and a "Name" column **224**. A record **226** for "John Smith" has the first name "John" in the "First Name" column **220** and the last name "Smith" in the column **222**. The name field **224** returns the text "The name is John Smith" by referencing the fields in brackets, according to the format <fieldRef field= 'Column Name'> as shown in column **224**.

FIG. **8***b* employs a variant of the referencing scheme illustrated in FIG. **8***a*. FIG. **8***a* illustrates a table **230** that includes a "First Name" column **232**, a "Last Name" column **234** and a "Name" column **236**. A record **238** for "John Smith" has the first name "John" in the "First Name" column **232** and the last name "Smith" in the column **234**. The name field **236** returns the text "The name is John Smith" by referencing the fields by defined variables 'fn' and 'ln' as shown in column **236**. The variables are defined according to the format variable :=fieldAt (parameter, 'Column Name') and the variables may be referenced in a return statement as shown in column **236**.

Record Contents

As previously described, a given row may contain values for any column. However, to determine all of the columns that might be used by a record would involve scanning every possible column. To avoid this problem, in the preferred embodiment, the table **100** illustrated in FIG. **3** includes a "RecordContents" column that indicates those columns within which a particular record has stored values.

FIG. **9** illustrates the table **100** with a "RecordContents" column **127** that includes pointers to the columns containing values for a particular record. For example, the "Record-Contents" column **127** for row **110** has pointers to the column **124** and a column **125** but does not have a pointer to the column **126** because the row **110** does not have a value for the column **126**. As previously described, since every column has a corresponding row that defines the column, the "RecordContents" column **127** has a defining row **129**. Like any cell, the cell containing the record contents can be versioned, providing the ability to do record versioning.

Folders

To provide increased efficiency in managing information, the table **100** includes a data type defined as a folder. FIG. **10** illustrates the structure of a folder. As illustrated in the Figure, the table **100** includes a "Parent Folder" column **240** and a "Folder Children" column **242**. A folder has a corresponding record. For example, a folder entitled "Contacts" has a corresponding row **244** as illustrated in FIG. **10**. The "Folder Children" column **242** of the "Contacts" folder includes pointers to those records that belong to the folder. Similarly, those records that belong to a folder include a pointer to that folder in the "Parent Folder" column **240**.

The folder structure illustrated in FIG. **10** facilitates searching. As previously described, a column may be searched according to a folder specified in the column definition. If a folder is searched, the system accesses the record corresponding to the folder and then searches all of the records pointed to by that folder.

Further, the synchronization feature described above may be used to generate the list of items in a folder. For example, in FIG. **10**, the 'Folder Parent' and 'Folder Children' columns may be synchronized. When the 'Folder Parent' field **240** for record **138** is set to reference the 'Contacts' folder represented by row **244**, the list of items in the 'Contacts' folder ('FolderChildren') is automatically updated to store a reciprocal reference to record represented by row **138** by including its OID, **1100**, in the "Folder Children" column **242**.

Text Indexing System

The present invention includes an indexing system that provides for rapid searching of text included in any cell in the table **100**. Each key phrase is extracted from a cell and stored in a list format according to a predefined hierarchy. For example, the list may be alphabetized, providing for very rapid searching of a particular name.

FIG. **11** illustrates the extraction of text from the table **100** to a list **250**. The list **250** is shown separately from the table **100** for purposes of illustration but, in the preferred embodiment, the list **250** comprises part of the table **100**. The list **250** stores cell identification numbers for each word in the list where a cell identification number may be of the format <record OID, column OID>. For example, the word "Ventura" occurs in cells **252**, **254** and **256** that correspond to different rows and different columns. The word "Ventura" in the list **250** contains a pointer, or cell identification number, to cells **252**, **254** and **256**.

Similarly, each cell stores the references to the key phrases within it using 'anchors'. As illustrated in FIG. **12**, an anchor contains a location (such as the start and stop offset within the text), and an identification number. Both the text and the anchor are stored in the cell **252**. Other kinds of domains also support anchors. For example, graphical images support the notion of 'hot spots' where the anchor position is a point on the image.

As previously described, each key phrase is stored as a record in the database and the OID of the record equals the identification number described with reference to FIG. **12**. One column stores the name of the key phrase and another stores the list of cell identification numbers that include that phrase. Key phrases may have comments of their own, which may also be indexed.

The sorted list **250** as illustrated in FIG. **11** is stored as a Folder, as illustrated in FIG. **13**. A cell identification field **274** maintains the cells that include the term corresponding to that record. The "Parent Folder" column **240** for each of the terms on the list **250** indicates that the parent folder is an index with a title "Natural." The "Natural" folder has a row **276** that has pointers in the "Folder Children" column **242** to all of the terms in the list **250**.

The "Natural" folder corresponds to an index sorted by a specific type of algorithm. Computer programs generally sort using a standard collating sequence such as ASCII. The present invention provides an improvement over this type of sorting and the improved sorting technique corresponds to the "Natural" folder. Records in the "Natural" folder are sorted according to the following rules:

1) A key phrase may occur at more than one point in the list. In particular:
   - 1a) Key phrases may be permuted and stored under each permutation. For example: 'John Smith' can be stored under 'John' and also under 'Smith'. Noise words such as 'a' and 'the' are ignored in the permutation.
   - 1b) Key phrases which are numeric or date oriented may be stored under each possible location. For example: '1984' can be stored under the digit '1984' and also under 'One thousand, nine hundred . . .', and 'nineteen eighty four'.

2) Numbers are sorted naturally. For example, '20' comes after '3' and before '100'.
3) Prefixes in key phrases are ignored. For example, 'The Big Oak' is sorted under 'Big'.
4) Key phrases are stemmed, so that 'Computers' and 'Computing' map to the identical key phrase record.

The preferred embodiment of the routine for generating positions for entering the key phrases into the 'Natural' folder is as follows:

1) Capitalize the key phrase to avoid case sensitivity problems. For example: 'John Smith the 1st' becomes 'JOHN SMITH THE 1ST'.
2) Each word in the key phrases is stemmed using standard techniques. Eg "COMPUTERS" becomes "COMPUT".
3) Permute the key phrase. This results in a new set of multiple key phrases based on the original key phrase. For example 'JOHN SMITH THE 1ST' produces the set {'JOHN SMITH THE 1ST'; 'SMITH THE 1ST JOHN'; 'THE 1ST JOHN SMITH'; '1ST JOHN SMITH THE'}.
4) Noise prefixes are eliminated. In the example above, the third entry, 'THE 1ST JOHN SMITH', is eliminated. If no phrases are left after elimination, the original phrase is used. For example, an entry for 'TO BE OR NOT TO BE" would be preserved even if all noise words were eliminated.
5) For each result, numbers and dates are expanded to all possible text representations, and text representations are converted to numeric. For example: '1ST JOHN SMITH THE' generates the set: {'1ST JOHN SMITH THE'; 'FIRST JOHN SMITH THE'}
6) Finally, each modified key phrase is used to determine the position of a reference to the main key phrase record, and an entry is made in the folder accordingly. For example, '1ST JOHN SMITH THE' is stored between '1' and '2', while 'FIRST JOHN SMIT THE' is stored after 'FIR' and before 'FIS.'

FIG. 22*a* illustrates the results of a prior art sorting algorithm while FIG. 22*b* illustrates the results of a sorting alogrithm according to the present invention.

Extracting the Key Phrases

To generate a sorted list, the system must first extract the key phrases or words from the applicable cells. The combination of structured information and text allows various combinations of key phrase extraction to be used. In full text extraction, every word is indexed, which is typical for standard text retrieval systems. In column extraction, the whole contents of the column are indexed which corresponds to a standard database system. According to a third type of extraction, automatic analysis, the contents of the text are analyzed and key phrases are extracted based on matching phrases, semantic context, and other factors. Finally, in manual selection extraction, the user or application explicitly marks the key phrase for indexing.

Date Indexing System

The date indexing scheme is very similar to the text indexing scheme as previously described. Important dates are extracted from the text and added to an 'Important Date' list. Each important date is represented by a 'Important Date' record. The 'Important Date' records are stored in a 'Important Dates' folder, which is sorted by date.

The important dates are extracted from the text. The system may search for numeric dates, such as '4/5/94' or date-oriented text, such as "Tomorrow", "next Tuesday" or "Christmas". FIG. 23 illustrates the correspondence between cells of the table of FIG. 2 and a sorted date index.

Important Date records are assigned special predetermined OIDS since they always have the same identity in any system. Assigning predetermined OID's to dates allows Important Dates to be shared across systems. The predetermined OID is generated by using a special session identification number that signifies that the OID is an Important Date. In this case, the timestamp represents the value of the Important Date itself, not the time that it was created.

Associative Queries

As previously described, a sorted key word list is generated from the text in cells and list stored in a folder whose records point to the text cells. The associations between the list of records with text and the list of key phrases is two-way since the cells that include text point to the key words. FIG. **14** illustrates this two way correspondence. Each record can point to multiple key phrases, and each key phrase can point to multiple records.

FIG. **15** is a graphical representation of the two way association between records and the key word list. Each record in a plurality of records **298** through **300** may point to one or more important word entries **310** through **312**. Similarly, each important word entry may point to one or more records. A single level search involves starting at one node (on either side of the graph) and following the links to the other side. For example, a user may wish to find the records including the word "Shasta." First, the important word index would be accessed to find the word "Shasta" and the records pointed to by this word would then be retrieved. This search is indicated by the arrows **314** and **316** where word "Shasta" corresponds to cell **318**. Similarly, a user may wish to locate all of the important words included in a particular record, indicated by the arrows **320** and **322** in FIG. **15**.

The search can be extended by repeatedly following the links back and forth to the desired level. FIG. **16***a* illustrates this concept. As an example, the term "Shasta" may correspond to a dog with extraordinary intelligence such that in one record, "Shasta" is described as a dog and another record, 'Shasta' is described as a genius. If the user wishes to find the words associated with 'Shasta', the system locates "Shasta" in the "Important Words" folder which points to the records including the word "Shasta." In turn, the records pointed to contain pointers to the "Important Words" list for each indexed word in the record. Since "Shasta" appears with "dog" and "genius" in the records, these words are retrieved by the system.

This type of searching may be extended indefinitely. FIG. **16***b* illustrates an additional level of searching. Continuing with the above example, the word "genius" may occur in records referring to Dirac, and the word "dog" associated with "Checkers," such that the multilevel search illustrated in FIG. **16***b* results in a retrieval of "Dirac" and "Checkers" when provided with the word "Shasta."

A relevance ranking can be created based on weights associated with each link and type of key word, and the records can be displayed in order of descending relevance. In the preferred embodiment, if two or more nodes are used as the starting point, the relevance is based on the distance from all nodes. In this way, only nodes which are near all the initial nodes will have a high relevance. Many other relevance rankings apart from distance may be used.

To refine the search, filters can be used to constrain the links that are followed. For example, the search may be filtered such that only the type "Person" is listed such that, in the above example, Shasta will be associated with Dirac but not Checkers.

Knowledge Base and Thesaurus

The present invention includes a knowledge base and thesaurus to further improve searching capabilities.

Each important word record (term) included within the thesaurus contains a pointer to a 'concept' record. Each concept record contains pointers to other concept records, and to the terms that are included within the bounds of that concept. FIG. **17** illustrates the structure of the thesaurus. The table **100** includes a "Parent Concept" column **352**, a "Concept Name" column **354**, a "Synonyms" column **356**, a "More Specific Terms" column **358**, a "More General Terms" column **360** and a "See Also" column **362**. A concept record **350** defines the concept "IBM" and the Synonyms column **356** points to records that are synonymous with IBM, a record **364** with a label field with the value "IBM" and a record **366** with a label field with the value "International Business Machines." The records **364** and **366** have pointers in the "parent concept" field that point to the parent concept record **350**.

The thesaurus structure illustrated in FIG. **17** provides for greater flexibility than exact synonyms. The "More Specific Terms" column **358** of the concept record **350** associated with "IBM" points to a concept record **368** associated with the IBM PC with an assigned weight of 100%, where the weight percentage reflects the similarity between the initial term "IBM" and the related term "IBM PC." Similarly, the "More General Terms" column **360** of the concept record **350** associated with "IBM" points to a concept record **372** associated with Computer Companies with an assigned weight of 60%. The "See also" column points to a record associated with the concept "Microsoft" with a weight of 70%, where the weight percentage reflects the similarity between the initial term "IBM" and the related term "IBM PC."

The Thesaurus illustrated in FIG. **17** enhances the searching mechanisms previously described with reference to FIGS. **14**–**16***b*. The system first locates the record associated with a key word and locates the parent concept record pointed to by the key word record. The system may then follow some or all of the pointers in the columns **356**, **358**, **360** and **352** and return of the OID's stored in the 'Concept Name' column **354**.

Since key phrases and concepts are stored as records in this system, any other columns may be used to extend the knowledge and information stored therein. In particular, through the use of OID's, the system can store any kind of relationship, including relationsihps other than thesaural relationships, between key phrases, concepts and other records.

Application Support

The database of the present invention has been described without reference to its interface with applications that may use the invention as their primary storage and retrieval system. As previously described with reference to FIG. **2**, the present database includes an interface to support applications programs. Components in the application support system include external document support, hypertext, document management and workflow, calendaring and scheduling, security and other features.

Further, the present invention includes various user interface components that allow have been developed to provide full access to the structure of the database of the present invention. In particular, a new kind of structured word processor will be presented. The Specification will describe each component of the application support system separately.

External Documents

The present invention supports indexing of external documents. The table **100** stores the filenames of documents, such as word processor documents, where the contents of the files are not directly stored in the database. The documents names may be stored in a column with a specialized "External Document" domain. The external documents may reside in the mass memory **32** or on a multi-source that interfaces with the system through device control **36**.

To index documents external to the table **100**, prior to processing, an external document is converted into a plain text format. Key phrases are then extracted as previously described. In particular, fields in the text can be determined and mapped to fields within the database. For example, a 'Memo' document may contain the text: 'To: John Smith. From: Mary Doe'. This text can be mapped to the fields called 'to' and 'from', and the values of these fields set accordingly. The analysis of the text in this way can be changed for different types of external documents such as memos, legal documents, spread sheets, computer source code and any other type of document. For each extracted key phrase, a start and stop point within the text is determined. A list of anchors of the format previously described, <start, stop, key phrase> is generated by the parser and stored within the table **100** under the external document domain.

Viewing External Documents

When a user views an external document on the display screen **37**, the stored anchors are overlaid on top of the document such that it appears that the external document has been marked with hypertext. When the user clicks the switches **45** or **47** of the mouse **50** on a section of the external document display, the corresponding anchor is determined from the various start and stop coordinates. The OID of the key phrase corresponding to the anchor is stored within the anchor, and can be used for the purposes of retrieving the key phrase record or initiating a query as previously described.

Dynamic Hypertext

The present invention supports Hypertext. Hypertext systems typically associate a region of text with a pointer to another record, as illustrated in FIG. **18**. This creates a 'hard-coded' link between the source and the target. When s user clicks on the source region, the target record is loaded and displayed. If the target record is absent, the hypertext jump will fail, possibly with serious consequences.

The present system uses a new approach based on a dynamic association between records. In the preferred embodiment, each hypertext region is associated with a key phrase, not a normal record. When the user clicks the switches **45** or **47** of the mouse **50** on the source region, all the records associated with the key phrase are retrieved and ranked using any of the associative search techniques previously described. As illustrated in FIG. **19**, the application can then display on the display screen **37** either the highest ranked item, or present all the retrieved items and allow the user to pick the one to access.

In certain applications, the user may want to access a single 'default' item. This item can be determined automatically, by picking the item at the top of the dynamically generated list, or manually, by letting the user pick the item explicitly and then preserving this choice in the anchor itself.

The Generic Word Processor

The database of the present invention includes a novel Structured Word Processor that may be used in conjunction with the table **100**.

The structured word processor of the present invention uses the "boxes and glue" paradigm introduced by Donald Knuth in T_EX. According to this paradigm, a page of text is created by starting with individual characters and concatenating the characters to form larger units, called "boxes," and then combining these boxes into yet larger boxes. FIG. **20***a* illustrates three character boxes **400**, **402** and **404** concatenated to form a word box **406**. FIG. **20***b* illustrates four word boxes **410**, **412**, **414** and the word box **406** combined to form a horizontal line box **408**. Horizontal boxes are used for words and other text tokens that are spaced horizontally inside another box, such as a line (or column width). FIG. **20***c* illustrates the combination of the horizontal line box **408** with another horizontal line box **4242** to form a vertical box **420**. Vertical boxes are used for paragraphs and other objects that are spaced vertically inside other boxes, such as page height.

Boxes may be attached to other boxes with "glue." The glue can stretch or shrink, as needed. For example, in a justified sentence, the white space between words is stretched to force the words to line up at the right edge of the column. Glue can be used for between-character (horizontal) spacing, between-word (horizontal) spacing including "tab" glue, that "sticks" to tab markings. Glue may also be used for between-line (vertical) spacing and between-paragraph (vertical) spacing.

When a record of the table **100** is edited, each word and field definition is converted into boxes. The system organizes these boxes into a tree structure of line boxes and paragraph boxes, as illustrated in FIG. **21**. Shown there is a record hierarchy **460**, corresponding to the hierarchy of a record, and a layout hierarchy **470**, corresponding to the hierarchy of a layout such as a document generated according to the word processor described with reference to FIGS. **20***a*–**20***c*. The record structure hierarchy **460** represents the record structure of the table **100** where a record **462** corresponds to a row in the table **100** and the record **462** includes a plurality of attributes, including attribute **464**, that correspond to the columns of the table **100**. In turn, the attribute may include a variety of items. For example, the attribute **464** includes text, represented by block **466**, field references represented by block **468** and other items as shown.

The layout hierarchy **470** comprises a document **472** which in turn comprises a plurality of pages, including page **474**. The page **474** comprises a plurality of paragraphs including paragraphs **430** and **431** and the paragraph **430** comprises a plurality of lines, including lines **432** and **434**. The paragraph **431** includes line **436**.

The word processor of the present invention allows the document **472** to be inserted into the record **462** by providing a plurality of boxes, including boxes **438**, **440** and **442**, common to both the record structure hierarchy **460** and the layout hierarchy **470**. For example, the box **438** corresponds to part of the line **432** and comprises part of the text of attribute **464** as illustrated by block **466**. Similarly, the box **440** corresponds to part of the line **434** and may comprise a field reference as indicated by block **468**. Thus, the shared box structure as illustrated in FIG. **21** allows any type of word processing document to interface with any record in the table **100**.

Conceptually, each box is kept as a bitmap, and its height and width are known, so the system displays the tree structure **450** by displaying all of the bitmaps corresponding to the boxes in the tree. If the tree is changed, for example, by adding a new word, only the new word box and a relatively small number of adjacent boxes need be recalculated. Similarly, line breaks or restructuring of a paragraph does not alter most of the word boxes, which may be reused, and only the lineboxes need be recalculated.

To edit the tree structure **450** as illustrated in FIG. **21**, a user may click a cursor on a part of the text. The system locates the word box or glue that is being edited by a recursively descending through the tree structure **450**.

The word processor supports multiple fonts and special effects such as subscripts, dropcaps and other features including graphic objects. A word in a different font than a base font is in a different box and may have a different height from other boxes on a line. The height of a linebox the height of the largest wordbox within it. Effects within a word can be handled by breaking a word into subboxes with no glue between them. Again, the height of a wordbox is the height of the largest box within it. Graphic objects, such as bitmaps, may be treated and formatted as a fixed width box.

The word processor of the present invention may be used to edit records in the table **100**. The text associated with each field in a record can be considered a "paragraph" for the purposes of inter-field spacing, text flow within a field, and other formatting parameters. Storing all the fields in the same way during text-editing allows the movement of text and "flow" to appear natural.

As previously described, the text being edited is divided into fields, with each field corresponding to a column in the underlying database. Unlike a traditional static data entry form, the positions and sizes of the attributes are not fixed but are dynamic and all the features of a word-processor such as fonts and embedded graphics are available to edit the record fields.

Similarly, all of the features of a database such as lookups and mailmerge are available to the word processor. All of the attributes that apply to data entry for a particular field are enforced by the word processor. Such attributes might include a mask (such as ###-####), existence requirements, range and value constraints, etc. The fields can be explicitly labelled, or hidden and implied.

The word processor of the present invention allows existing fields to be added by typing the prefix of a field name and pressing a button. The system then completes the rest of the field name automatically.

The word processor of the present invention supports other database features. For example, new fields can be created by a user by using a popup dialog box. Similarly, references to other records or important words can be added by a dialog box. With particular regard to the table **100** of the present invention, OID references may support fields within other fields and a particular field within other fields supports the use of 'templates,' where a template is a list of field references embedded in text. For example, the template "Enter the first name here <fieldref id =firstName> and the last name here <fieldref id=lastName>" would appear to the user as "Enter the first name here: John and the last name here: Doe." Templates allow a user to build dynamic forms quickly and easily without having to use complicated form drawing tools.

The user interface for the word processor of the present invention allows a user to switch between two modes of data entry. The word-processor of the present invention is used for flexible entry into one record at a time, while a columnar view is used for entering data in columns. The user can switch back and forth between these two views with no loss of data and switching from the word processor to the columnar view will cause the fields that were entered in the single item to become the columns to be displayed in the columnar view.

Finally, the 'fields within fields' that are apparent in the word processor view become separated into columns in a columnar view. The user can then make changes in columnar mode, and then, when switching back to the word processor view, the columns become combined once again.

Passwords

It is often required that access to particular data items be restricted to certain users. In order to apply these restrictions, an information management system must determine the identity of the user requesting access. This is currently done in two ways, physically measuring a unique quality of the uses of requesting information from the user, most current information management systems rely on the second approach, by using 'passwords'. However, to avoid security problems with a password system, three guidelines are applied to passwords:

a) the password should not be made of common words, because an aggressor can use a brute force approach and a dictionary to guess the password;

b) the password should be longer rather than shorter; and

c) the password should be changed often, so that even if is stolen it will not be valid for long.

Finally, a password should never be written down or embedded into a login script and should always be interactive.

According to the present password system, a user's identity is determined through an extensive question and answer session. The responses to certain personal questions very quickly identify the user with high accuracy. Even an accurate mimic will eventually fail to answer correctly if the question and answer session is prolonged.

For example, sample questions might be: 'What is your favorite breakfast cereal?'; 'Where were you in April 1990?' 'What color is your toothbrush?'. These questions are wide ranging and hard to mimic. Furthermore, the correct responses are natural English sentences, with an extremely large solution space, so that a brute force approach is unlikely to be successful.

To improve the effectiveness of the response, an exact matching of user response and stored answer is not required and 'fuzzy' and 'associative' matching can be used according to the synonym, thesaurus and other features of the present invention.

According to the password system of the present invention, the user creates the list of questions and corresponding answers, which are then stored. Because the user has complete control over the questions, the user may find the process of creating the questions and answers enjoyable, and as a result, change the questions and answer list more frequently, further enhancing system security.

According to the preferred embodiment, a user creates a list of 50–100 questions and answers that are encrypted and stored. The questions can be entirely new, or can be based on a large database of interesting questions. When the user logs on the system, the system randomly selects one of the questions related to that user and presents the question to the user. The user then types in a response, which is matched against the correct answer. The matching can be fuzzy and associative, as described above. If the response matches correctly, access is allowed.

In an alternate embodiment, more security may be provided by repeatedly asking questions until a certain risk threshold is reached. For example, if the answer to 'What color is your toothbrush?' is the single word 'Red', then brute force guessing may be effective in this one case. In this scenario, repeatedly asking questions will diminish the probability of brute force success.

Summary

While the invention has been described in conjunction with the preferred embodiment, it is evident that numerous alternatives, modifications, variations and uses will be apparent to those skilled in the art in light of the foregoing description. Many other adaptations of the present invention are possible.

We claim:

1. A data storage and retrieval system for a computer memory, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row having an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column having an OID to identify each said logical column; and wherein

at least one of said logical rows has an OID equal to the OID of a corresponding one of said logical columns, and at least one of said logical rows includes logical column information defining each of said logical columns.

2. The system of claim **1** wherein said logical column information defines one of said logical columns to contain information for enabling determination of OIDs from text entry.

3. The system of claim **1** wherein said one of said logical columns contains information including a search path that references a folder, said folder including a group of logical rows of a similar type.

4. The system of claim **1** wherein:

said logical column information defines one of said logical columns to contain information for synchronizing two logical columns reciprocally.

5. The system of claim **4** wherein said one of said logical columns contains information including reciprocal pointers to said two logical columns.

6. The system of claim **1** wherein:

at least one of said plurality of logical rows includes information defining the type of a different logical row; and

at least one of said plurality of logical rows includes a logical cell that contains a pointer to said logical row including logical row type information.

7. The system of claim **1** wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

8. The system of claim **1** wherein at least one of said logical rows is a folder type logical row, said folder type logical row including at least one logical cell that contains data and a plurality of pointers to a plurality of other logical rows included within said folder.

9. The system of claim **8** wherein said plurality of other logical rows included within said folder each includes a logical cell that contains a pointer to said folder type logical row.

10. The system of claim **1** wherein said OID's are variable length and include data related to a session identification number and a timestamp.

11. A data storage and retrieval system for a computer memory, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said plurality of logical rows contains a logical cell that contains a pointer to a different logical row and at least one of said plurality of logical rows includes information defining the type of a different logical row; and

means for searching said table for said pointer.

12. A data storage and retrieval system for a computer memory, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said logical rows contains a logical cell that contains a pointer to a different logical row and at least one of said logical rows includes logical column information defining each of said logical columns; and

means for searching said table for said pointer.

13. The system of claim 12 wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

14. The system of claim 12 wherein at least one of said logical rows is a folder type logical row, said folder type logical row including at least one logical cell that contains data and a plurality of pointers to a plurality of other logical rows included within said folder.

15. The system of claim 14 wherein said plurality of other logical rows included within said folder each includes a logical cell that contains a pointer to said folder type logical row.

16. A data storage and retrieval system for a computer memory, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information; and

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column, wherein said OID's are variable length.

17. A data storage and retrieval system for a computer memory, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and

means for indexing data stored in said table.

18. The system of claim 17 wherein said means for indexing further comprises:

means for searching a plurality of logical cells within said table for a key word, said means capable of searching a logical column containing unstructured text and a logical column containing structured data; and

means for inserting a logical row corresponding to said key word into said table.

19. The system of claim 18 wherein:

said inserted logical row includes a logical cell that contains a pointer to a searched logical cell that contains the keyword corresponding to said inserted logical row; and

said searched logical cell that contains a keyword corresponding to said inserted logical row contains a pointer to said inserted logical row.

20. The system of claim 19 wherein said pointer to said searched logical cell includes the OID's of the logical column and logical row defining said searched logical cell.

21. The system of claim 19 wherein said searched logical cell includes an anchor that marks said key word.

22. The system of claim 17 wherein one of said plurality of logical rows of said table includes a folder type logical row that includes at least one pointer to said key word.

23. The system of claim 18 wherein said searching means further includes:

means for searching for every word in a text logical cell;

means for searching for every entry in a logical column;

means for searching for data based on automatic analysis; and

means for searching for data marked by a user.

24. A data storage and retrieval system for a computer memory, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said logical cells includes a pointer to an index record; and

means for indexing data stored in said table.

25. The system of claim 24 wherein said indexing means further comprises:

means for searching said table for a key word; and

means for creating an index record for said key word, said index record including one or more pointers to a logical cell in said table that contains said key word.

26. The system of claim 25 further including querying means, said querying means further including:

means for locating said index record according to the query of a user;

means for retrieving at least one logical cell in said table pointed to by said located index record.

27. The system of claim **26** wherein said index locating means includes means for locating said index record pointed to by said at least one retrieved logical cell.

28. The system of claim **27** wherein said index locating means and said record retrieval means each includes weighing means for weighing key words and retrieved logical cells according to pre-defined search criteria.

29. The system of claim **27** wherein said index locating means and said record retrieval means each includes filtering means for filtering key words and retrieved logical cells according to pre-defined search criteria.

30. The system of claim **25** wherein said indexing means further includes means for indexing external documents.

31. A method for storing and retrieving data in a computer memory, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said logical rows has an OID equal to the OID to a corresponding one of said logical columns, and at least one of said logical rows includes logical column information defining each of said logical columns.

32. The method of claim **31** wherein said logical column information defines one of said logical columns to contain information for enabling determination of OIDs from text entry.

33. The method of claim **31** wherein one of said logical columns contains information including a search path that references a folder, said folder including a group of logical rows of a similar type.

34. The method of claim **31** wherein:

said logical column information defines one of said logical columns to contain information for synchronizing two logical columns reciprocally.

35. The method of claim **34** wherein said one of said logical columns contains information including reciprocal pointers to said two logical columns.

36. The method of claim **31** wherein:

at least one of said plurality of logical rows includes information defining the type of a different logical row; and

at least one of said plurality of logical rows includes a logical cell that contains a pointer to said logical row including logical row type information.

37. The method of claim **31** wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

38. The method of claim **37** wherein at least one of said logical rows is a folder type logical row, said folder type logical row including at least one logical cell that contains data and a plurality of pointers to a plurality of other logical rows included within said folder.

39. The method of claim **38** wherein said plurality of other logical rows included within said folder each includes a logical cell that contains a pointer to said folder type logical row.

40. The method of claim **31** wherein said OID's are variable length and include data related to a session identification number and a timestamp.

41. A method for storing and retrieving data in a computer memory, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said logical rows contains a logical cell that contains a pointer to a different logical row and at least one of said plurality of logical rows includes information defining the type of a different logical row; and

means for searching said table for said pointer.

42. A method for storing and retrieving data in a computer memory, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said logical rows contains a logical cell that contains a pointer to a different logical row and at least one of said logical rows includes logical column information defining each of said logical column; and

searching said table for said pointer.

43. The method of claim **42** wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

44. The method of claim **42** wherein at least one of said logical rows is a folder type logical row, said folder type logical row including at least one logical cell that contains data and a plurality of pointers to a plurality of other logical rows included within said folder.

45. The method of claim **44** wherein said plurality of other logical rows included within said folder each includes a logical cell that contains a pointer to said folder type logical row.

46. A method for storing and retrieving data in a computer memory, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information; and

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column, wherein said OID's are variable length.

47. A method for storing and retrieving data in a computer memory, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and

indexing data stored in said table.

48. The method of claim **47** wherein said step of indexing data further comprises the steps of:

searching a plurality of logical cells within said table for a key word, said searching element capable of searching a logical column containing unstructured text and a logical column containing structured data; and

inserting a logical row corresponding to said key words into said table.

49. The method of claim **48** wherein:

said inserted logical row includes a logical cell that contains a pointer to a searched logical cell that contains the keyword corresponding to said inserted logical row; and

said searched logical cell that contains a keyword corresponding to said inserted logical row contains a pointer to said inserted logical row.

50. The method of claim **49** wherein said pointer to said searched logical cell includes the OID's of the logical column and logical row defining said searched logical cell.

51. The method of claim **49** wherein said searched logical cell includes an anchor that marks said key word.

52. The method of claim **48** wherein one of said plurality of logical rows of said table includes a folder type logical row that includes at least one pointer to said key word.

53. The method of claim **48** wherein said step of searching a plurality of cells within said table for a key word further comprises the steps of:

searching for every word in a text logical cell;

searching for every entry in a logical column;

searching for data based on automatic analysis; and

searching for data marked by a user.

54. A method for storing and retrieving data in a computer memory, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of logical rows, each said logical row including an object identification number (OID) to identify each said logical row, each said logical row corresponding to a record of information;

a plurality of logical columns intersecting said plurality of logical rows to define a plurality of logical cells, each said logical column including an OID to identify each said logical column; and wherein

at least one of said logical cells includes a pointer to an index record; and

indexing data stored in said table.

55. The method of claim **54** wherein said step of indexing data further comprises the steps of:

searching said table for a key word; and

creating an index record for said key word, said index record having one or more pointers to a logical cell in said table that contains said key word.

56. The method of claim **55** further comprising the steps of:

locating said index record according to the query of a user;

retrieving at least one logical cell in said table pointed to by said located index record.

57. The method of claim **56** wherein said step of locating said index record further comprises:

locating said index record pointed to by said at least one retrieved logical cell.

58. The method of claim **57** wherein said step of locating said index record further comprises:

weighing key words and retrieved logical cells according to pre-defined search criteria.

59. The method of claim **57** wherein said step of locating said index record further comprises:

filtering key words and retrieved logical cells according to pre-defined search criteria.

60. The method of claim **54** wherein said step of indexing data further comprises the step of:

indexing external documents.

* * * * *

’775 Patent

US006163775A

United States Patent [19]

Wlaschin et al.

[11] **Patent Number: 6,163,775**

[45] **Date of Patent: *Dec. 19, 2000**

[54] **METHOD AND APPARATUS CONFIGURED ACCORDING TO A LOGICAL TABLE HAVING CELL AND ATTRIBUTES CONTAINING ADDRESS SEGMENTS**

[75] Inventors: **Scott Wlaschin**; **Robert M. Gordon**, both of Los Angeles; **Louise J. Wannier**, La Canada, all of Calif.; **Clay Gordon**, New York, N.Y.

[73] Assignee: **Enfish, Inc.**, Pasadena, Calif.

[*] Notice: This patent is subject to a terminal disclaimer.

[21] Appl. No.: **09/035,187**

[22] Filed: **Mar. 5, 1998**

Related U.S. Application Data

[63] Continuation of application No. 08/383,752, Mar. 28, 1995, Pat. No. 5,729,730.

[51] **Int. Cl.**[7] **G06F 17/30**

[52] **U.S. Cl.** **707/3**; 707/1; 707/4; 707/100

[58] **Field of Search** 707/3, 4, 1, 100

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,201,046 4/1993 Goldberg et al. 707/1
5,295,256 3/1994 Bapat 707/500
5,305,389 4/1994 Palmer 382/1
5,359,724 10/1994 Earle 395/425
5,375,237 12/1994 Tamaka et al. 395/650
5,421,012 5/1995 Khoyi et al. 395/650
5,459,860 10/1995 Burnett et al. 707/100
5,537,591 7/1996 Oka 707/100
5,537,633 7/1996 Suzuki et al. 707/100
5,553,218 9/1996 Li et al. 395/148
5,557,787 9/1996 Shin et al. 707/3
5,560,005 9/1996 Hoover et al. 707/3
5,564,046 10/1996 Nemoto et al. 707/1
5,729,730 3/1998 Wlaschin et al. 707/3

Primary Examiner—Thomas G. Black
Assistant Examiner—Frantz Coby
Attorney, Agent, or Firm—Morrison & Foerster, LLP

[57] **ABSTRACT**

The information management and database system of the present invention comprises a flexible, self-referential table that stores data. The table of the present invention may store any type of data, both structured and unstructured, and provides an interface to other application programs. The table of the present invention comprises a plurality of rows and columns. Each row has an object identification number (OID) and each column also has an OID. A row corresponds to a record and a column corresponds to a field such that the intersection of a row and a column comprises a cell that may contain data for a particular record related to a particular field, a cell may also point to another record. To enhance searching and to provide for synchronization between columns, columns are entered as rows in the table and the record corresponding to a column contains various information about the column. The table includes an index structure for extended queries.

60 Claims, 17 Drawing Sheets

120 122 130 124 134 126 132 100

108	OBJECT ID	TYPE [# 101]	[#1012] LABEL	ADDRESS [#1013]	EMPLOYED BY [#1019]	TITLE [#1033]	AUTHOR [#1032]
110	#1100	#1020 [COMPANY]	DEXIS	117 EAST COLORADO		N/A	N/A
138	#1101	#1010 [PERSON]	SCOTT WLASCHIN		#1100 [DEXIS]	N/A	N/A
	#1118	#1030 [BOOK]					#1122
	#1122	#1050 [MEMO]					#1122
	#1127	#1060 [DOCUMENT]		C:\WORD\ PROJ.DOC		PROJECT PLAN	#1101
136	#1019	# 210 [FIELD]	EMPLOYED BY				
135	# 210	# 111 [TYPE]	COLUMN				
140	# 111	# 111 [TYPE]	TYPE				

133

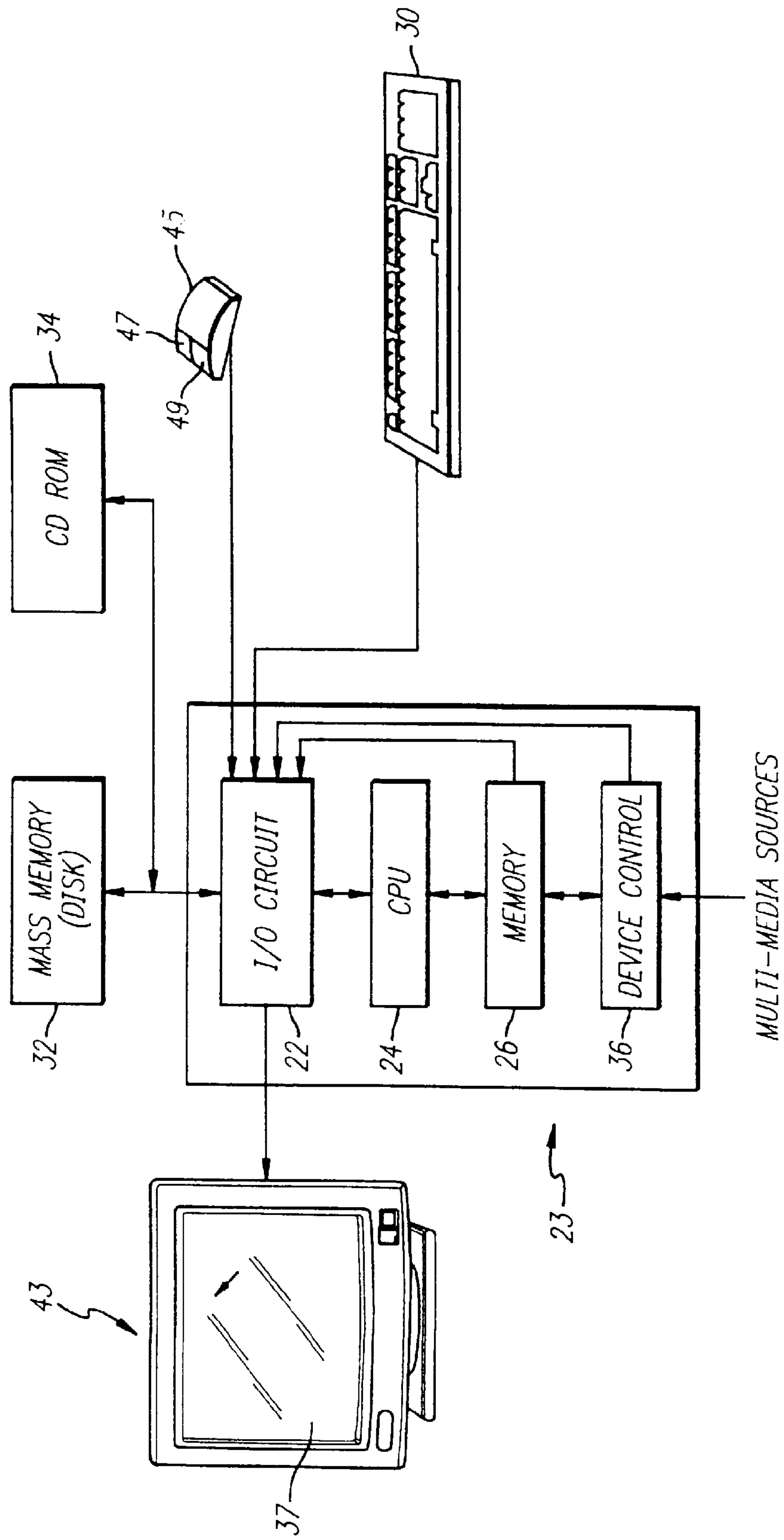

30
45
47
49
34
CD ROM
MASS MEMORY (DISK)
I/O CIRCUIT
CPU
MEMORY
DEVICE CONTROL
MULTI-MEDIA SOURCES
32
22
24
26
36
23
43
37

FIG. 1

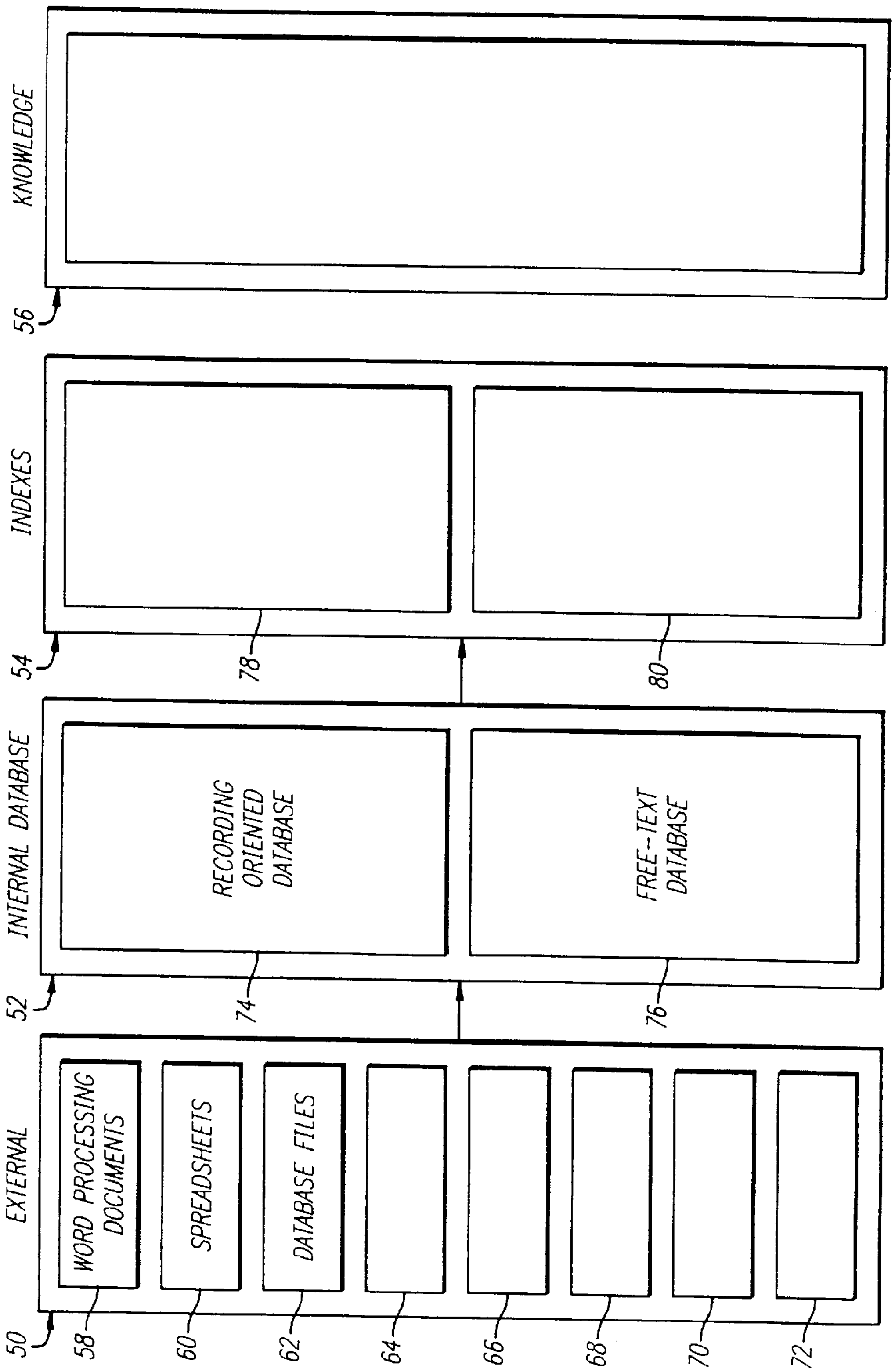

KNOWLEDGE
56
INDEXES
54
78
80
INTERNAL DATABASE
52
RECORDING ORIENTED DATABASE
74
FREE-TEXT DATABASE
76
EXTERNAL
50
WORD PROCESSING DOCUMENTS
58
SPREADSHEETS
60
DATABASE FILES
62
64
66
68
70
72

FIG. 2

FIG. 3

OBJECT ID	TYPE [# 101]	[#1012] LABEL	ADDRESS [#1013]	EMPLOYED BY [#1019]	TITLE [#1033]	AUTHOR [#1032]
#1100	#1020 [COMPANY]	DEXIS	117 EAST COLORADO		N/A	N/A
#1101	#1010 [PERSON]	SCOTT WLASCHIN		#1100 [DEXIS]	N/A	N/A
#1118	#1030 [BOOK]					#1122
#1122	#1050 [MEMO]					#1122
#1127	#1060 [DOCUMENT]		C:\WORD\PROJ.DOC		PROJECT PLAN	#1101
#1019	# 210 [FIELD]	EMPLOYED BY				
# 210	# 111 {TYPE]	COLUMN				
# 111	# 111 [TYPE]	TYPE				

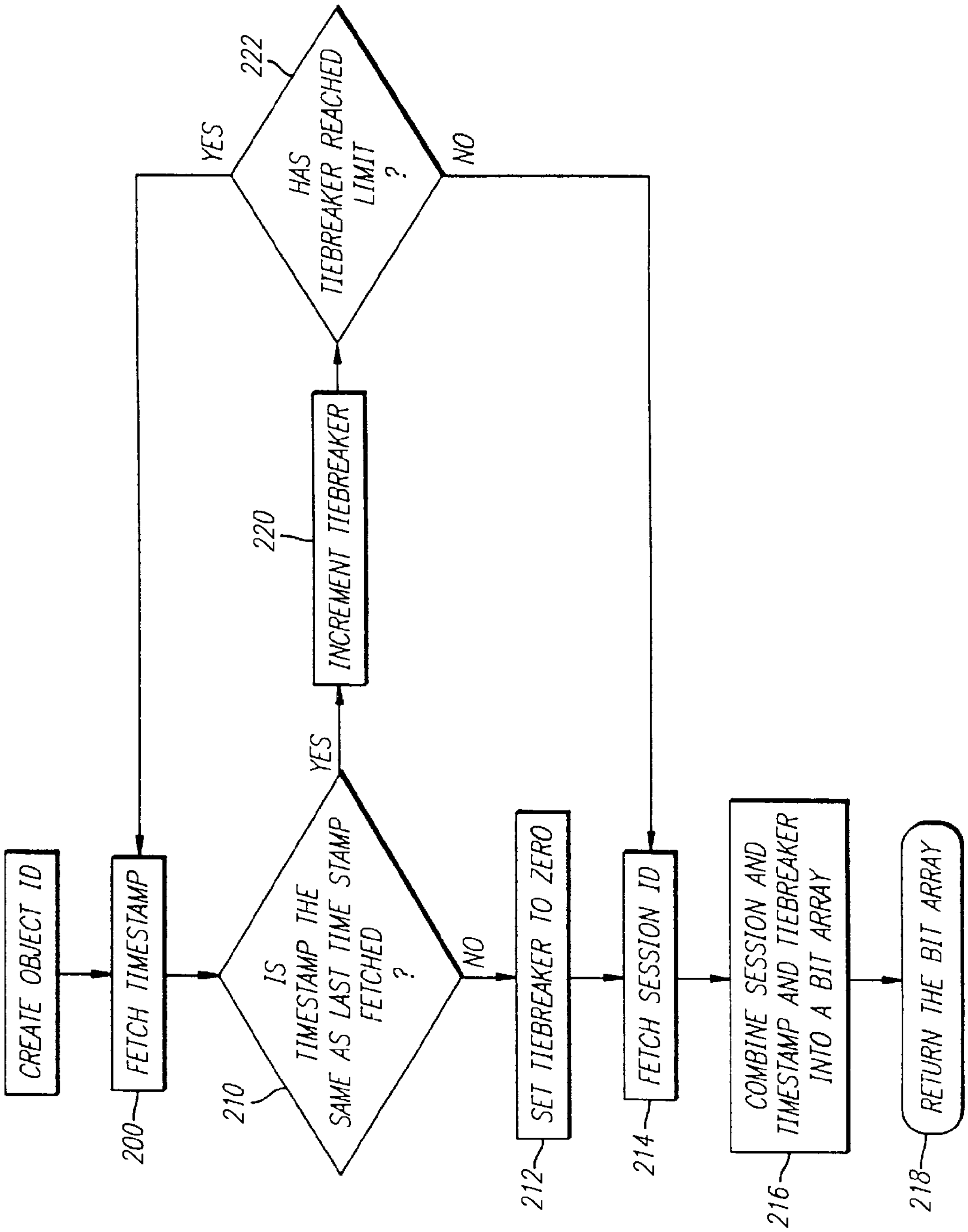
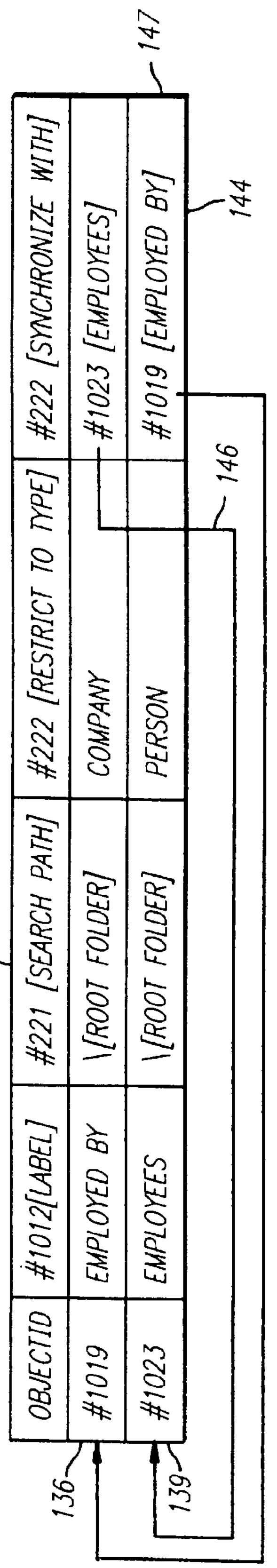
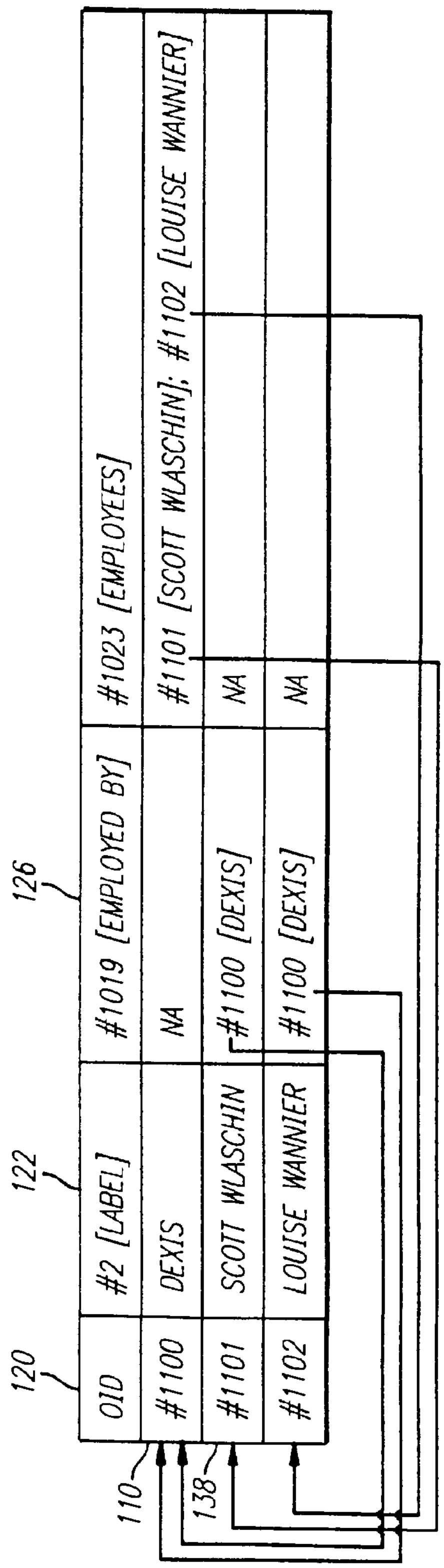

CREATE OBJECT ID
FETCH TIMESTAMP
200
IS TIMESTAMP THE SAME AS LAST TIME STAMP FETCHED ?
210
YES
NO
INCREMENT TIEBREAKER
220
HAS TIEBREAKER REACHED LIMIT ?
222
YES
NO
SET TIEBREAKER TO ZERO
212
FETCH SESSION ID
214
COMBINE SESSION AND TIMESTAMP AND TIEBREAKER INTO A BIT ARRAY
216
RETURN THE BIT ARRAY
218

FIG. 4



OBJECTID
#1012[LABEL]
#221 [SEARCH PATH]
#222 [RESTRICT TO TYPE]
#222 [SYNCHRONIZE WITH]
#1019
EMPLOYED BY
\[ROOT FOLDER]
COMPANY
#1023 [EMPLOYEES]
#1023
EMPLOYEES
\[ROOT FOLDER]
PERSON
#1019 [EMPLOYED BY]
147
144
146
146
136
139

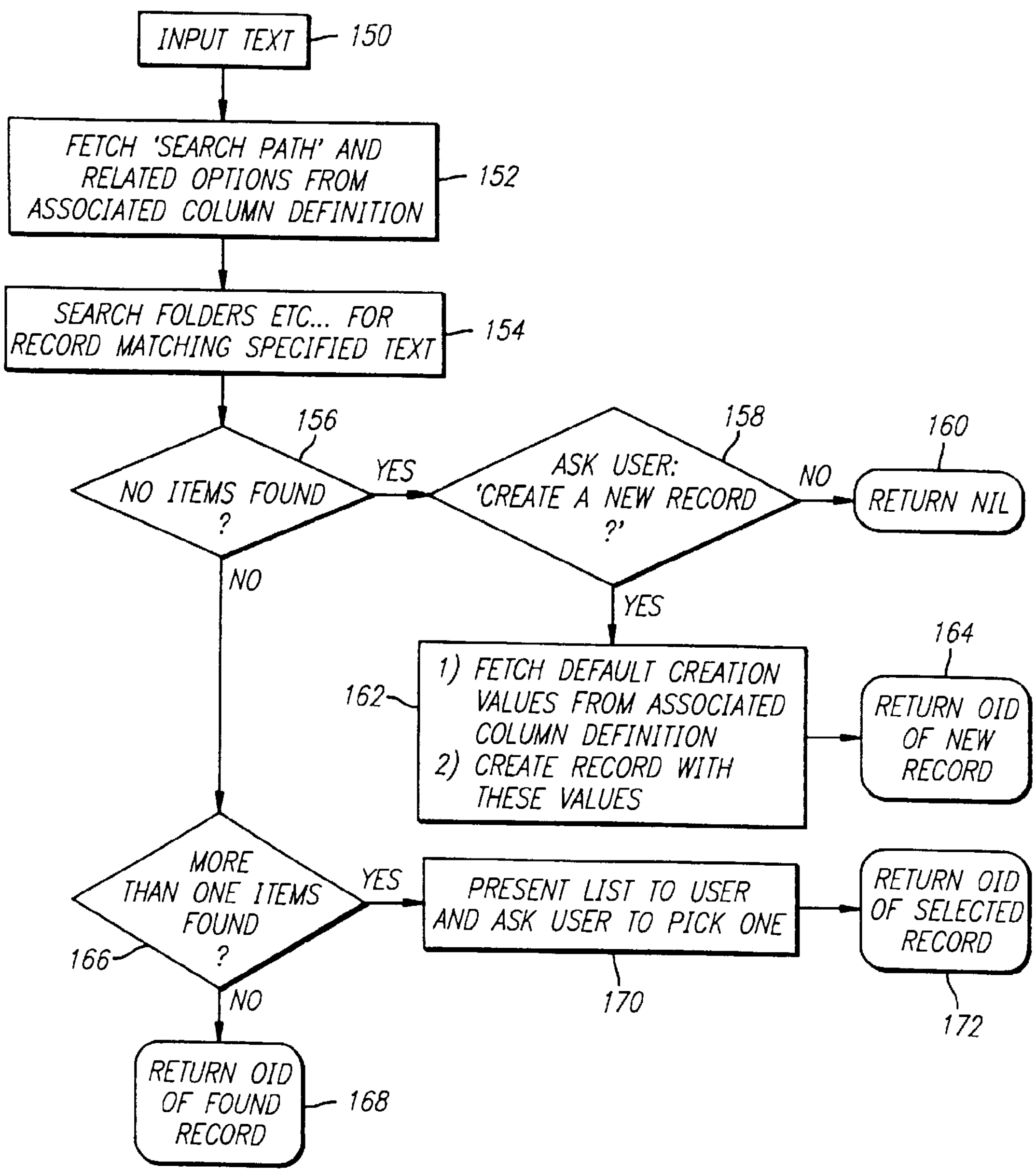
FIG. 5

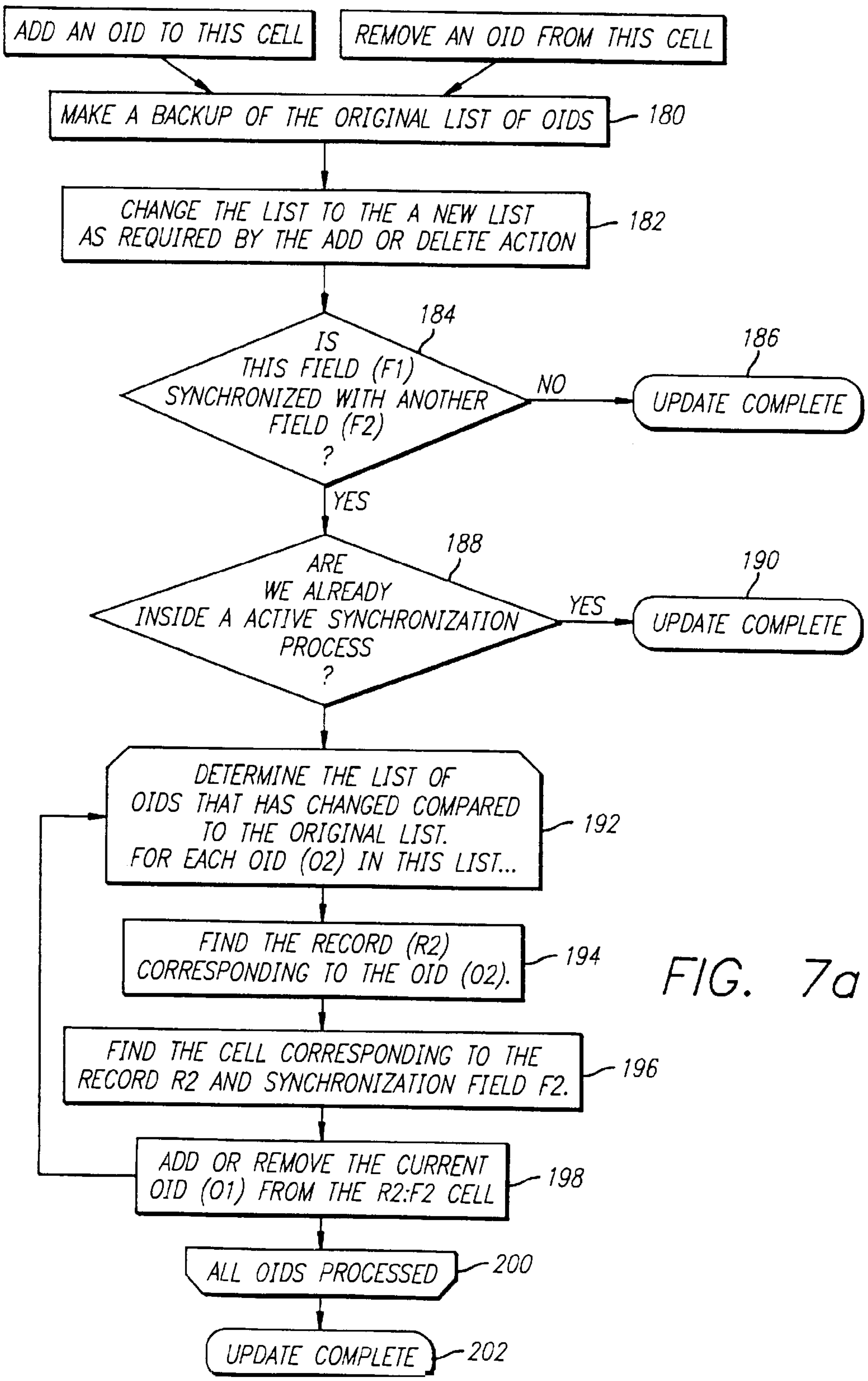


OID
#2 [LABEL]
#1019 [EMPLOYED BY]
#1023 [EMPLOYEES]
#1100
DEXIS
NA
#1101 [SCOTT WLASCHIN]; #1102 [LOUISE WANNIER]
#1101
SCOTT WLASCHIN
#1100 [DEXIS]
NA
#1102
LOUISE WANNIER
#1100 [DEXIS]
NA
120
122
126
110
138

FIG. 7b

FIG. 6


INPUT TEXT
150
FETCH 'SEARCH PATH' AND RELATED OPTIONS FROM ASSOCIATED COLUMN DEFINITION
152
SEARCH FOLDERS ETC... FOR RECORD MATCHING SPECIFIED TEXT
154
156
NO ITEMS FOUND ?
YES
158
ASK USER: 'CREATE A NEW RECORD ?'
NO
160
RETURN NIL
YES
NO
162
1) FETCH DEFAULT CREATION VALUES FROM ASSOCIATED COLUMN DEFINITION
2) CREATE RECORD WITH THESE VALUES
164
RETURN OID OF NEW RECORD
MORE THAN ONE ITEMS FOUND ?
166
YES
PRESENT LIST TO USER AND ASK USER TO PICK ONE
170
RETURN OID OF SELECTED RECORD
172
NO
RETURN OID OF FOUND RECORD
168



ADD AN OID TO THIS CELL
REMOVE AN OID FROM THIS CELL
MAKE A BACKUP OF THE ORIGINAL LIST OF OIDS
180
CHANGE THE LIST TO THE A NEW LIST AS REQUIRED BY THE ADD OR DELETE ACTION
182
184
IS THIS FIELD (F1) SYNCHRONIZED WITH ANOTHER FIELD (F2) ?
NO
186
UPDATE COMPLETE
YES
188
ARE WE ALREADY INSIDE A ACTIVE SYNCHRONIZATION PROCESS ?
YES
190
UPDATE COMPLETE
DETERMINE THE LIST OF OIDS THAT HAS CHANGED COMPARED TO THE ORIGINAL LIST. FOR EACH OID (02) IN THIS LIST...
192
FIND THE RECORD (R2) CORRESPONDING TO THE OID (02).
194
FIND THE CELL CORRESPONDING TO THE RECORD R2 AND SYNCHRONIZATION FIELD F2.
196
ADD OR REMOVE THE CURRENT OID (01) FROM THE R2:F2 CELL
198
ALL OIDS PROCESSED
200
UPDATE COMPLETE
202

FIG. 7a

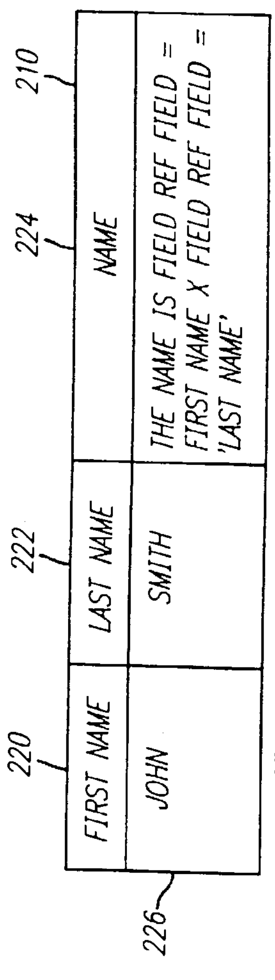


FIG. 8a

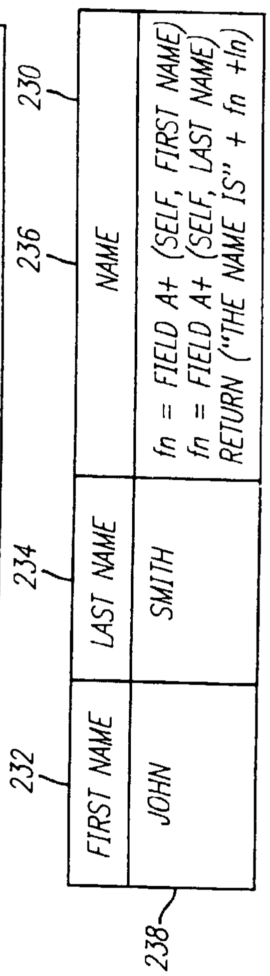


FIG. 8b

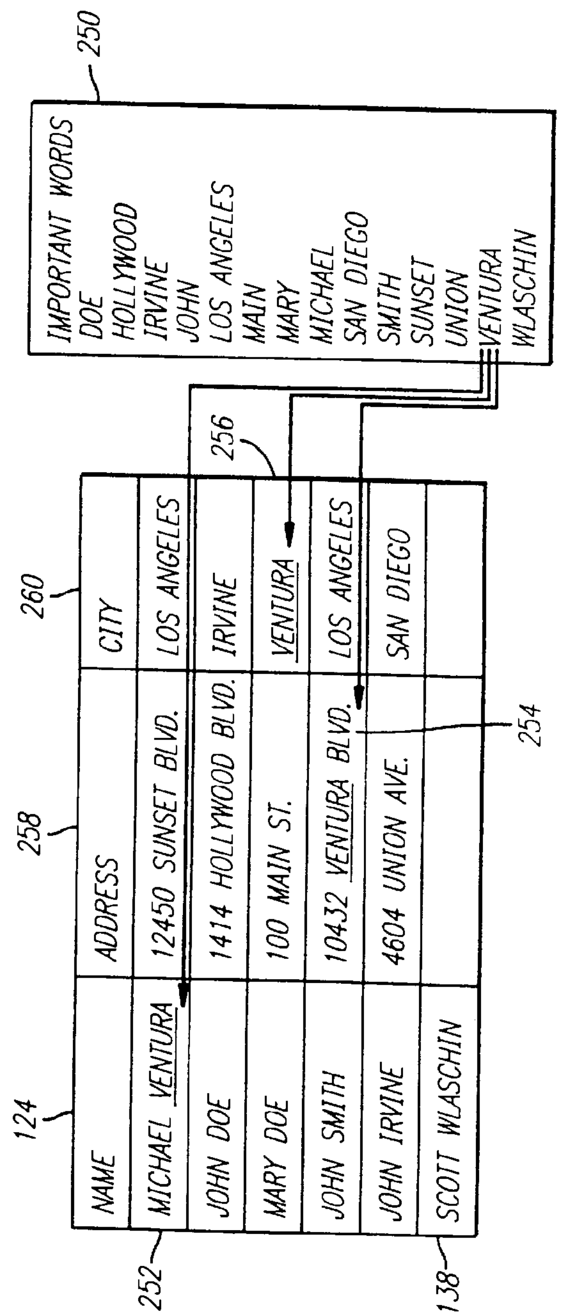


FIG. 11

FIG. 9

DEFINITION FOR RECORDCONTENTS COLUMN

RECORDCONTENTS COLUMN IN USE

100

OID	#101 [TYPE]	#2 [LABEL]	#1013 [ADDRESS]	#1019 [EMPLOYED BY]	#801 [RECORD CONTENTS]
#80	COLUMN	RECORD CONTENTS	NA	NA	
#1100	COMPANY	DEXIS	117 E COLORADO	NA	2 [LABEL]; 101 TYPE; 301 [PARENT FOLDER]; 1022 [COMPANY]; 1023 [TYPE OF BUSINESS]; 1013 [ADDRESS]; 1014 [CITY]; 1015 [STATE]; 1017 [PHONE]; 4 [COMMENT].
#1101	PERSON	SCOTT WLASCHIN	777 N CHESTER	#1100	2 [LABEL]; 101 TYPE; 301 [PARENT FOLDER]; 1012 [NAME]; 1013 [ADDRESS]; 1014 [CITY]; 1015 [STATE]; 4 [COMMENT]; 1019 [EMPLOYED BY]
#1118	BOOK	1984	NA	NA	2 [LABEL]; 101 TYPE; 301 [PARENT FOLDER]; 1033 [TITLE]; 1032 [AUTHOR]; 4 [COMMENT]

120 129 110 138

122 124 125 126 127

FIG. 10

DEFINITIONS FOR FOLDER RELATED COLUMNS

OBJECTID	#101 [TYPE]	#2 [LABEL]	#301 [PARENT FOLDER]	#320 [FOLDERCHILDREN]
#301	FIELD	PARENT FOLDER	NA	NA
#320	FIELD	FOLDERCHILDREN	NA	NA
#1100	COMPANY	DEXIS	#1070 [CONTACTS]	NA
#1101	PERSON	SCOTT WLASCHIN	#1070 [CONTACTS]	NA
#1070	FOLDER	CONTACTS	NA	1100 [DEXIS]; 1101 [SCOTT WLASCHIN]; 1102 [LOUISE WANNIER]; ETC.

FOLDERCHILDREN IN USE AS AN ATTRIBUTE OF A FOLDER OBJECT

138

244

240

242

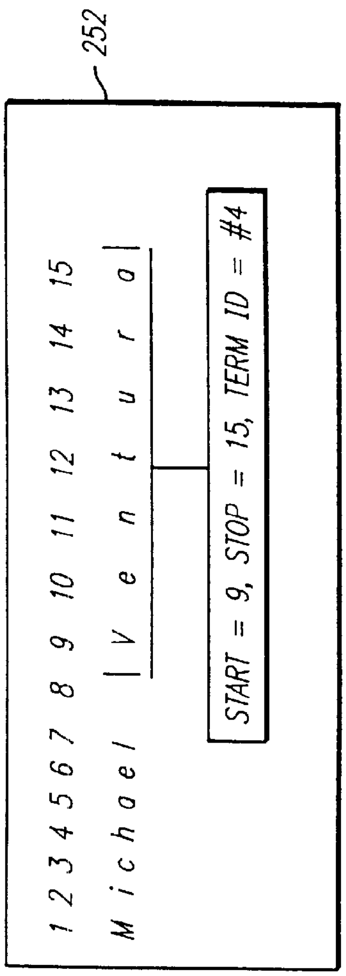

252
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
M i c h a e l | V e n t u r a |
START = 9, STOP = 15, TERM ID = #4

FIG. 12

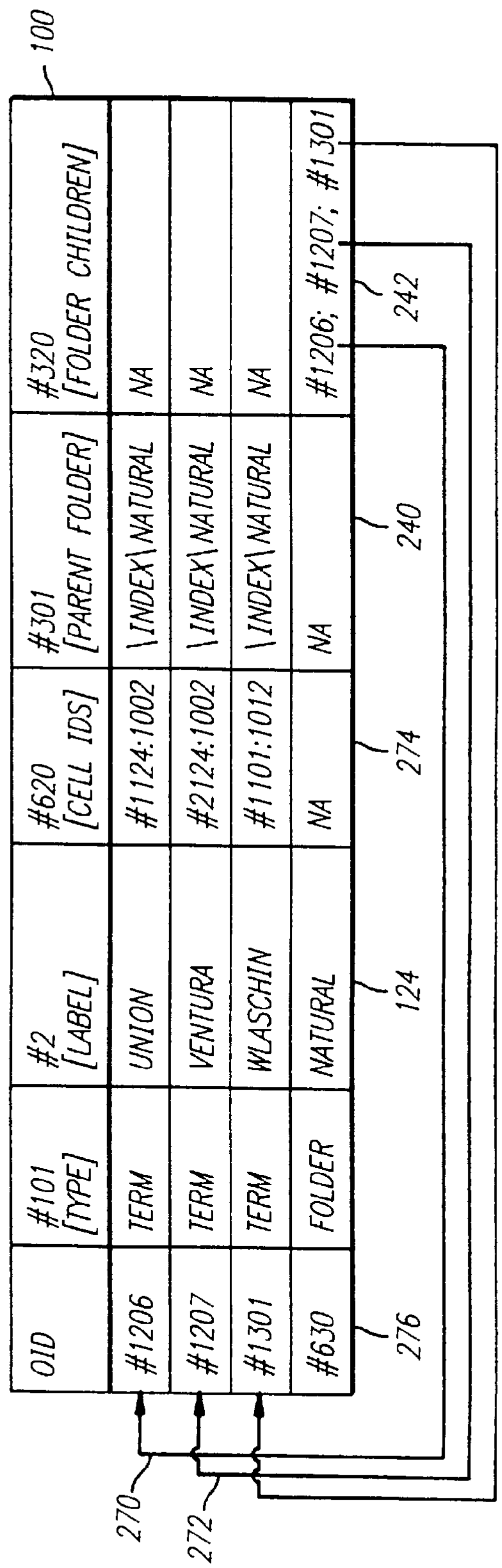

100
OID
#101 [TYPE]
#2 [LABEL]
#620 [CELL IDS]
#301 [PARENT FOLDER]
#320 [FOLDER CHILDREN]
#1206 TERM UNION #1124:1002 \INDEX\NATURAL NA
#1207 TERM VENTURA #2124:1002 \INDEX\NATURAL NA
#1301 TERM WLASCHIN #1101:1012 \INDEX\NATURAL NA
#630 FOLDER NATURAL NA NA #1206; #1207; #1301
276
124
274
240
242
270
272

FIG. 13

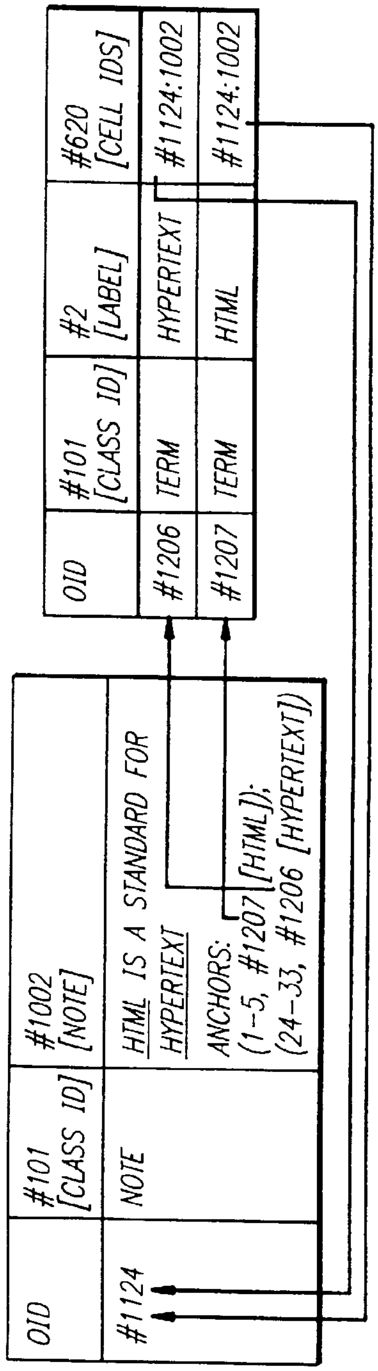

OID
#101 [CLASS ID]
#2 [LABEL]
#620 [CELL IDS]
#1206
TERM
HYPERTEXT
#1124:1002
#1207
TERM
HTML
#1124:1002
OID
#101 [CLASS ID]
#1002 [NOTE]
#1124
NOTE
HTML IS A STANDARD FOR HYPERTEXT
ANCHORS:
(1–5, #1207 [HTML]);
(24–33, #1206 [HYPERTEXT])

FIG. 14

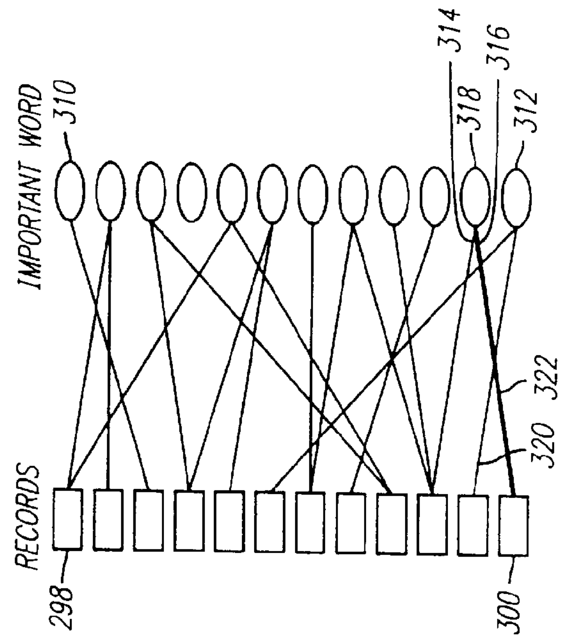

IMPORTANT WORD
310
314
316
318
312
RECORDS
298
300
320
322

FIG. 15

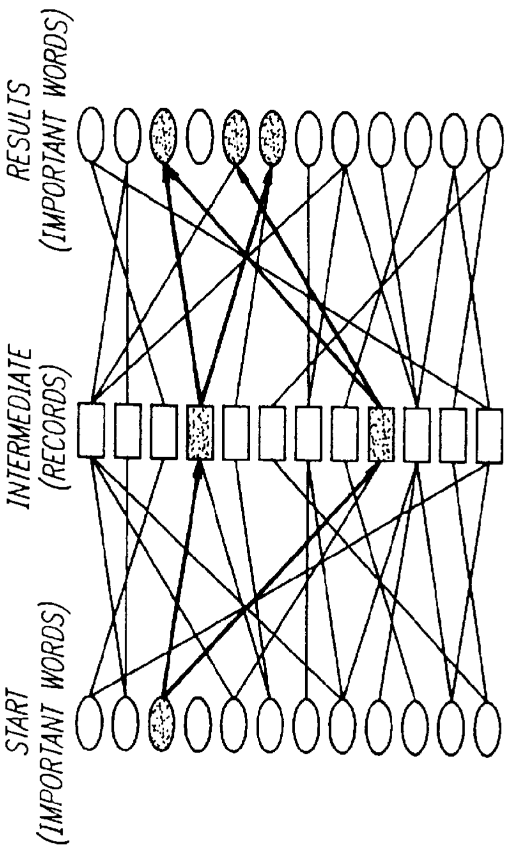


FIG. 16a

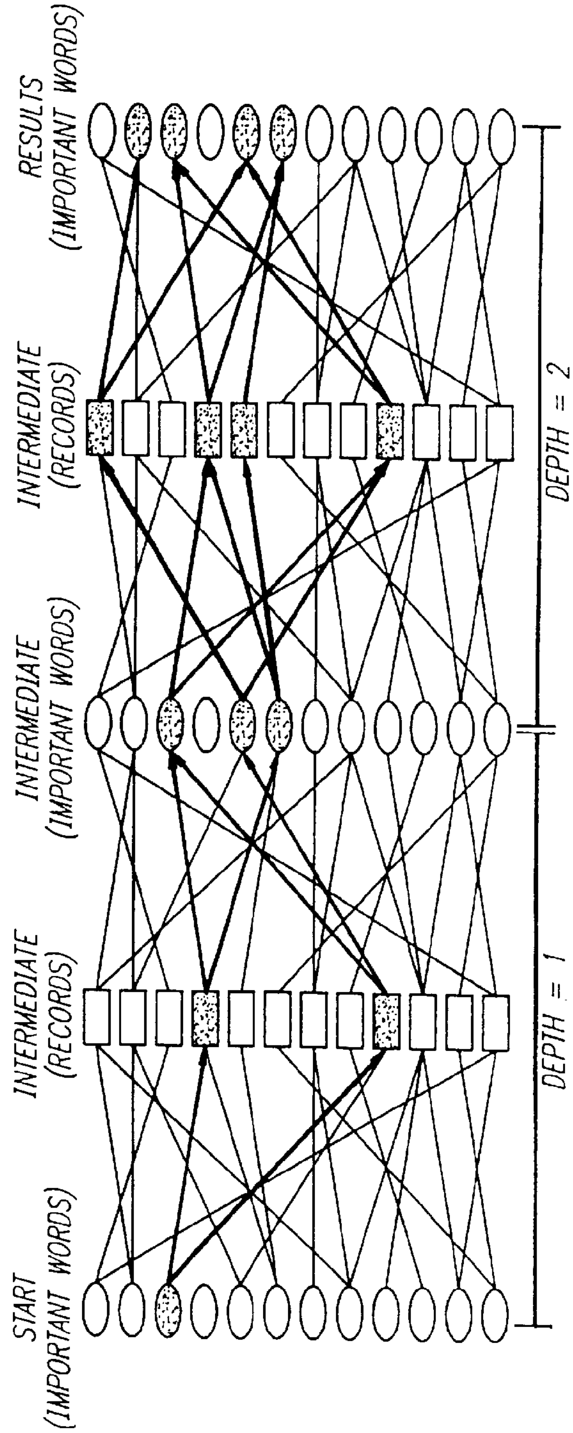


FIG. 16b

FIG. 17

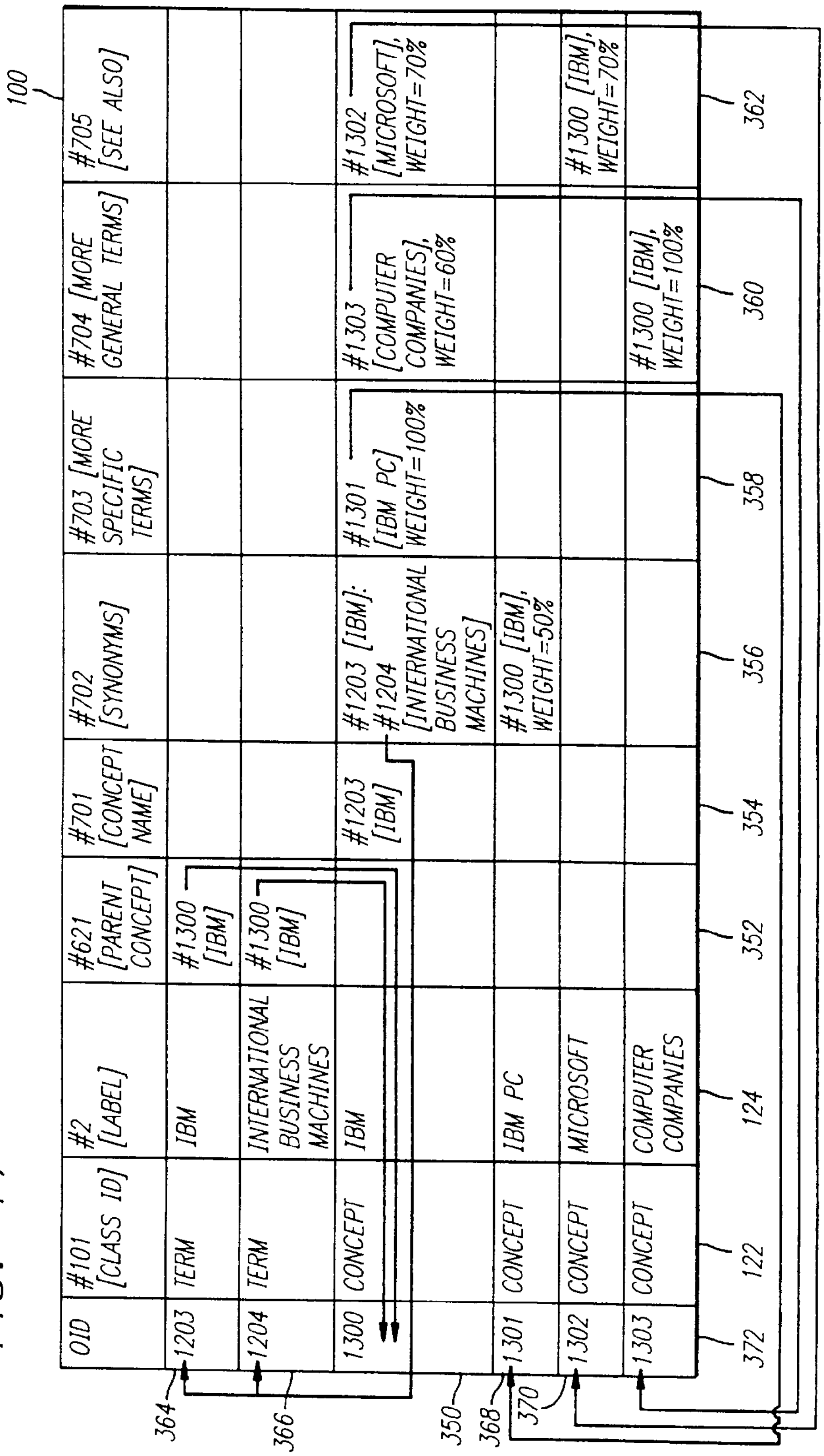
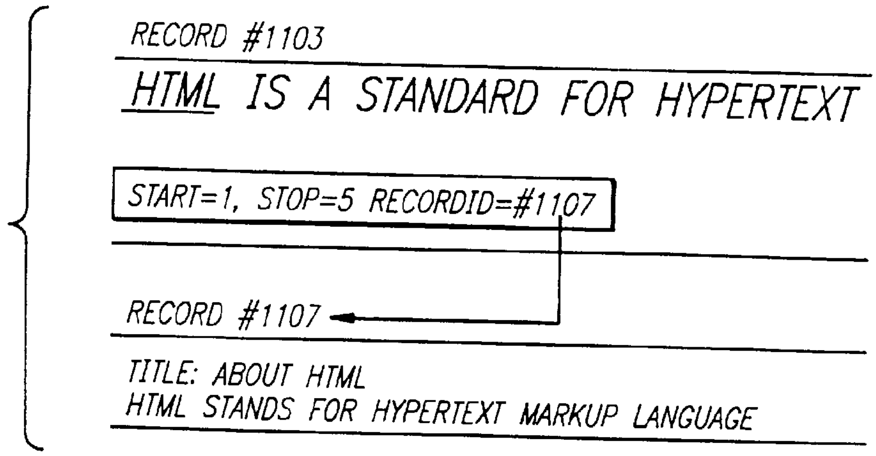

100
OID
#101 [CLASS ID]
#2 [LABEL]
#621 [PARENT CONCEPT]
#701 [CONCEPT NAME]
#702 [SYNONYMS]
#703 [MORE SPECIFIC TERMS]
#704 [MORE GENERAL TERMS]
#705 [SEE ALSO]
1203
TERM
IBM
#1300 [IBM]
1204
TERM
INTERNATIONAL BUSINESS MACHINES
#1300 [IBM]
1300
CONCEPT
IBM
#1203 [IBM]
#1203 [IBM]: #1204 [INTERNATIONAL BUSINESS MACHINES]
#1301 [IBM PC] WEIGHT=100%
#1303 [COMPUTER COMPANIES], WEIGHT=60%
#1302 [MICROSOFT], WEIGHT=70%
1301
CONCEPT
IBM PC
#1300 [IBM], WEIGHT=50%
1302
CONCEPT
MICROSOFT
#1300 [IBM], WEIGHT=70%
1303
CONCEPT
COMPUTER COMPANIES
#1300 [IBM], WEIGHT=100%
372
122
124
352
354
356
358
360
362
364
366
350
368
370

FIG. 18 PRIOR ART



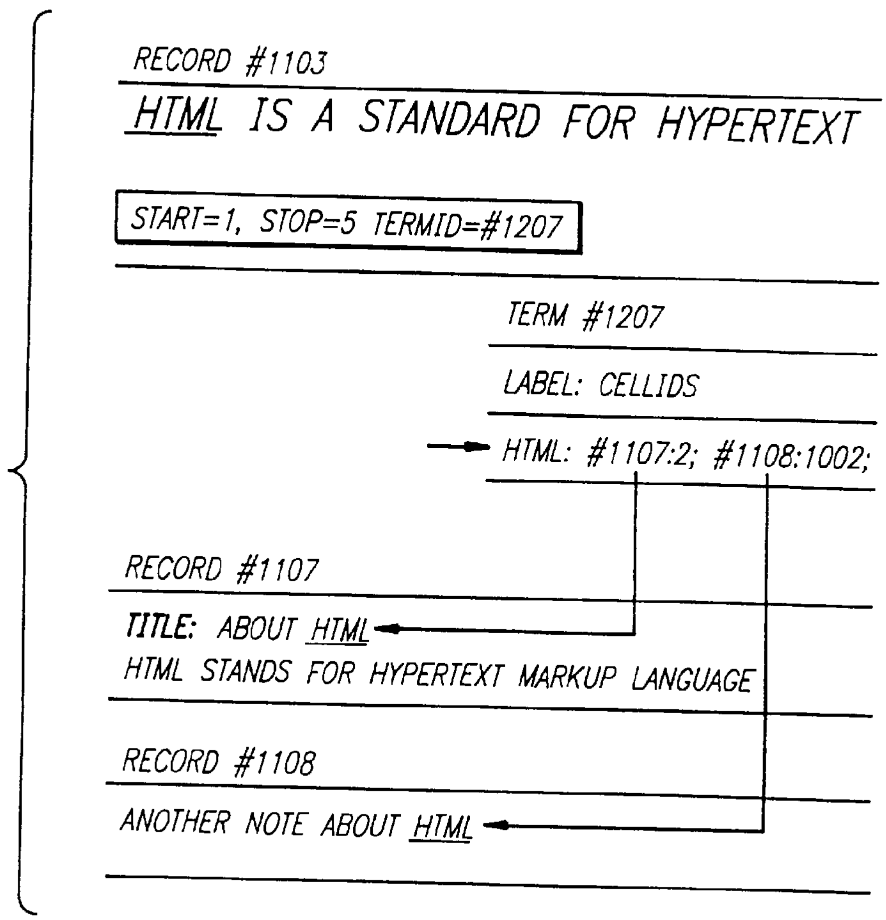


FIG. 19

FIG. 20a PRIOR ART

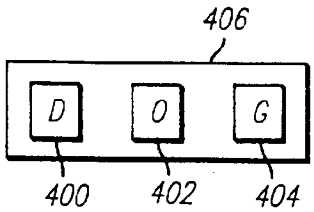


FIG. 20b PRIOR ART

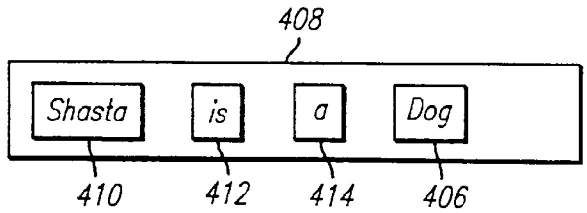


FIG. 20c PRIOR ART

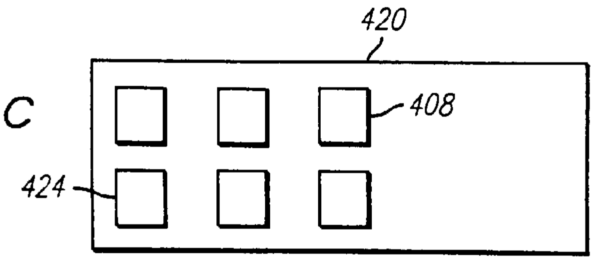


FIG. 21

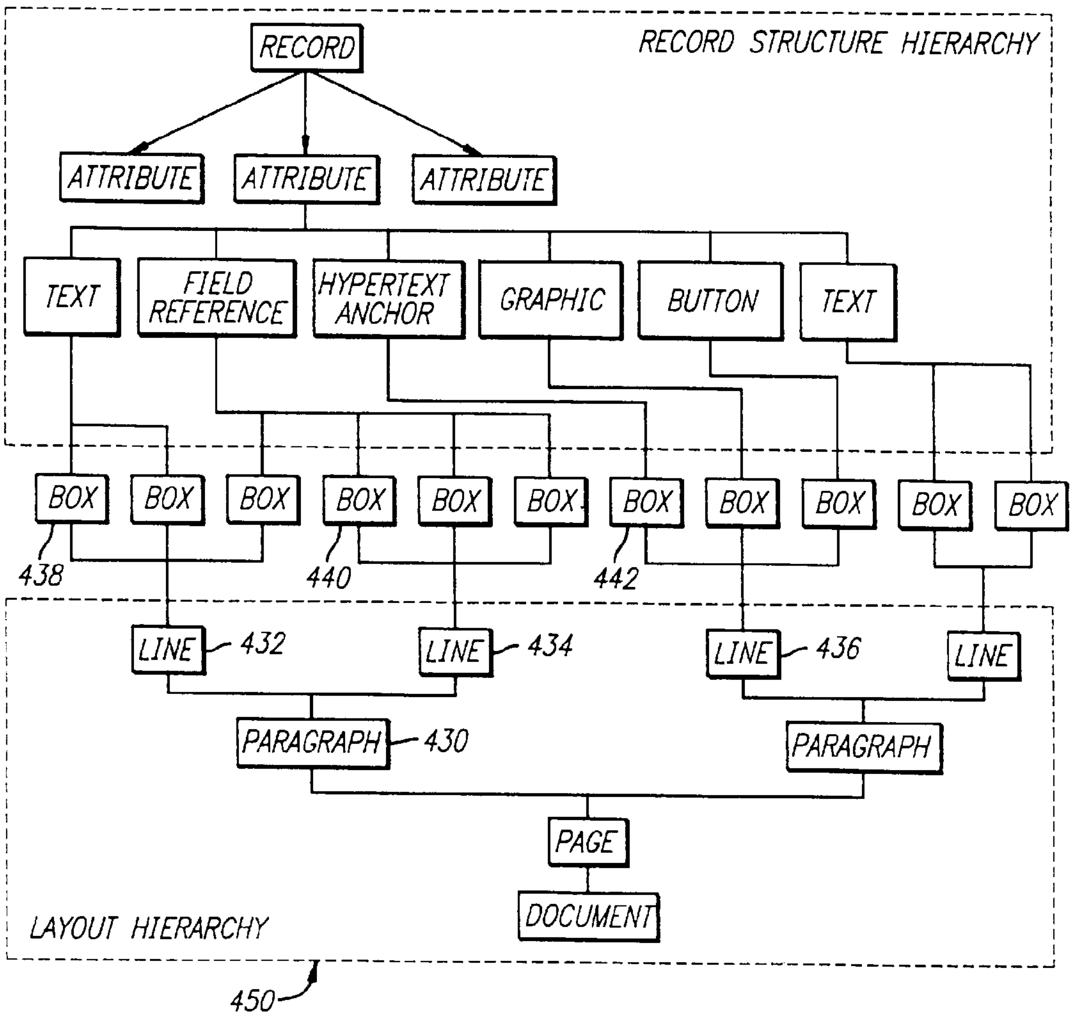


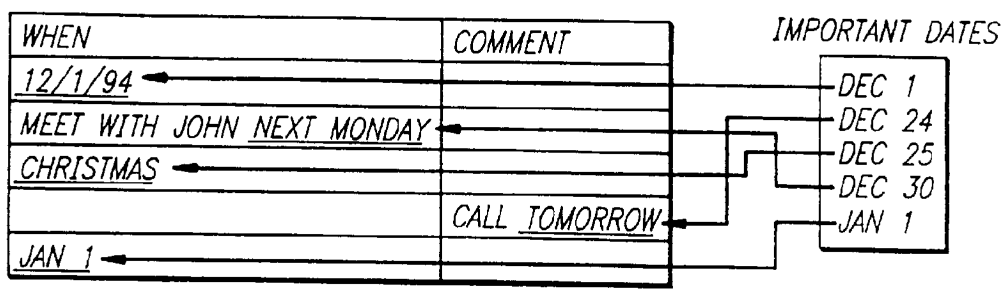
FIG. 22a
PRIOR ART

KEY PHRASE	SORTED UNDER
100	'1'
1984	'1', THEN '9'
20	'2', THEN '0'
3	'3'
JOHN SMITH	'J'
THE BIG OAK	'T'

FIG. 22b

KEY PHRASE	SORTED UNDER
3	3
20	20
100	100
1984	1984
THE BIG OAK	'B'–BIG
JOHN SMITH	'J'–JOHN
1984	'N'–NINETEEN EIGHTY FOUR
THE BIG OAK	'O'–OAK
100	'O'–ONE HUNDRED
1984	'O'–ONE THOUSAND NINE HUNDRED...
JOHN SMITH	'S'–SMITH
3	'T'–THREE
20	'T'–TWENTY
2	'T'–TWO

FIG. 23



METHOD AND APPARATUS CONFIGURED ACCORDING TO A LOGICAL TABLE HAVING CELL AND ATTRIBUTES CONTAINING ADDRESS SEGMENTS

RELATED APPLICATIONS

This is a continuation of application Ser. No. 08/383,752, filed Mar. 28, 1995, now U.S. Pat. No. 5,729,730, issued Mar. 17, 1998.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to a method and apparatus for storing, retrieving, and distributing various kinds of data, and more particularly, to an improved database architecture and method for using the same.

2. Art Background

Over the past 30 years, computers have become increasingly important in storing and managing information. During this time, many database products have been developed to allow users to store and manipulate information and to search for desired information. The continuing growth of the information industry creates a demand for more powerful databases.

The database products have evolved over time. Initially, databases comprised a simple "flat file" with an associated index. Application programs, as opposed to the database program itself, managed the relationships between these files and a user typically performed queries entirely at the application program level. The introduction of relational database systems shifted many tasks from applications programs to database programs. The currently existing database management systems comprise two main types, those that follow the relational model and those that follow the object oriented model.

The relational model sets out a number of rules and guidelines for organizing data items, such as data normalization. A relational database management system (RDBMS) is a system that adheres to these rules. RDBMS databases require that each data item be uniquely classified as a particular instance of a 'relation'. Each set of relations is stored in a distinct 'table'. Each row in the table represents a particular data item, and each column represents an attribute that is shared over all data items in that table.

The pure relational model places number of restrictions on data items. For example, each data item cannot have attributes other than those columns described for the table. Further, an item cannot point directly to another item. Instead, 'primary keys' (unique identifiers) must be used to reference other items. Typically, these restrictions cause RDBMS databases to include a large number of tables that require a relatively large amount of time to search. Further, the number of tables occupies a large amount of computer memory.

The object oriented database model, derived from the object-oriented programming model, is an alternative to the relational model. Like the relational model, each data item must be classified uniquely as belonging to a single class, which defines its attributes. Key features of the object-oriented model are: 1) each item has a unique system-generated object identification number that can be used for exact retrieval; 2) different types of data items can be stored together; and 3) predefined functions or behavior can be created and stored with a data item.

Apart from the limitations previously described, both the relational and object oriented models share important limitations with regard to data structures and searching. Both models require data to be input according to a defined field structure and thus do not completely support full text data entry. Although some databases allow records to include a text field, such text fields are not easily searched. The structural requirements of current databases require a programmer to predefine a structure and subsequent date entry must conform to that structure. This is inefficient where it is difficult to determine the structure of the data that will be entered into a database.

Conversely, word and image processors that allow unstructured data entry do not provide efficient data retrieval mechanisms and a separate text retrieval or data management tool is required to retrieve data. Thus, the current information management systems do not provide the capability of integrating full text or graphics data entry with the searching mechanisms of a database.

The separation of database from other programs such as word processors has created a large amount of text and other files that cannot be integrated with current databases. Various database, spreadsheet, image, word processing, electronic mail and other types of files may not currently be accessed in a single database that contains all of this information. Various programs provide integration between spreadsheet, word processing and database programs but, as previously described, current databases do not support effective searching in unstructured files.

The present invention overcomes the limitations of both the relational database model and object oriented database model by providing a database with increased flexibility, faster search times and smaller memory requirements and that supports text attributes. Further, the database of the present invention does not require a programmer to preconfigure a structure to which a user must adapt data entry. Many algorithms and techniques are required by applications that deal with these kinds of information. The present invention provides for the integration, into a single database engine, of support for these techniques, and shifts the programming from the application to the database, as will be described below. The present invention also provides for the integration, into a single database, of preexisting source files developed under various types of application programs such as other databases, spreadsheets and word processing programs. In addition, the present invention allows users to control all of the data that are relevant to them without sacrificing the security needs of a centralized data repository.

SUMMARY OF THE INVENTION

The present invention improves upon prior art information search and retrieval systems by employing a flexible, self-referential table to store data. The table of the present invention may store any type of data, both structured and unstructured, and provides an interface to other application programs such as word processors that allows for integration of all the data for such application programs into a single database. The present invention also supports a variety of other features including hypertext.

The table of the present invention comprises a plurality of rows and columns. Each row has an object identification number (OID) and each column also has an OID. A row corresponds to a record and a column corresponds to an attribute such that the intersection of a row and a column comprises a cell that may contain data for a particular record related to a particular attribute. A cell may also point to another record. To enhance searching and to provide for synchronization between columns, columns are entered as

rows in the table and the record corresponding to a column contains various information about the column. This renders the table self referential and provides numerous advantages, as will be discussed in this Specification.

The present invention includes an index structure to allow for rapid searches. Text from each cell is stored in a key word index which itself is stored in the table. The text cells include pointers to the entries in the key word index and the key word index contains pointers to the cells. This two way association provides for extended queries. The invention further includes weights and filters for such extended queries.

The present invention includes a thesaurus and knowledge base that enhances indexed searches. The thesaurus is stored in the table and allows a user to search for synonyms and concepts and also provides a weighting mechanism to rank the relevance of retrieved records.

An application support layer includes a word processor, a password system, hypertext and other functions. The novel word processor of the present invention is integrated with the table of the present invention to allow cells to be edited with the word processor. In addition, the table may be interfaced with external documents which allows a user to retrieve data from external documents according to the enhanced retrieval system of the present invention.

These and numerous other advantages of the present invention will be apparent from the following description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a functional block diagram illustrating one possible computer system incorporating the teachings of the present invention.

FIG. **2** is a block diagram illustrating the main components of the present invention.

FIG. **3** illustrates the table structure of the database of the present invention.

FIG. **4** is a flow chart for a method of computing object identification numbers (OID's) that define rows and columns in the table of FIG. **1**.

FIG. **5** is a part of the table of FIG. **2** illustrating the column synchronization feature of the present invention.

FIG. **6** is a flow chart for a method of searching the table of FIG. **2**.

FIG. **7***a* is a flow chart for synchronizing columns of the table of FIG. **2**.

FIG. **7***b* illustrates the results of column synchronization.

FIG. **8***a* illustrates a reference within one column to another column.

FIG. **8***b* illustrates an alternate embodiment for referring to another column within a column.

FIG. **9** illustrates a "Record Contents" column of the present invention that indicates which columns of a particular record have values.

FIG. **10** illustrates a folder structure that organizes records. The folder structure is stored within the table of FIG. **2**.

FIG. **11** illustrates the correspondence between cells of the table of FIG. **2** and a sorted key word index.

FIG. **12** illustrate the "anchors" within a cell that relate a word in a cell to a key word index record.

FIG. **13** illustrates key word index records stored in the table of FIG. **2**.

FIG. **14** illustrates the relationship between certain data records and key word index records.

FIG. **15** illustrates the relationship of FIG. **14** in graphical form.

FIG. **16***a* illustrates an extended search in graphical form.

FIG. **16***b* illustrates a further extended search in graphical form.

FIG. **17** illustrates the thesaurus structure of the present invention stored in the table of FIG. **2**.

FIG. **18** illustrates prior art hypertext.

FIG. **19** illustrates the hypertext features of the present invention.

FIG. **20***a* illustrates a character and word box structure of the word processor of the present invention.

FIG. **20***b* illustrates the word and horizontal line box structure of the word processor of the present invention.

FIG. **20***c* illustrates the vertical box structure of the word processor of the present invention.

FIG. **21** illustrates the box tree structure of the word processor of the present invention.

FIG. **22***a* illustrates the results of a prior art sorting algorithm.

FIG. **22***b* illustrates the results of a sorting algorithm according to the present invention.

FIG. **23** illustrates the correspondence between cells of the table of FIG. **2** and a sorted date index.

NOTATION AND NOMENCLATURE

The detailed descriptions which follow are presented largely in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein which form part of the present invention; the operations are machine operations. Useful machines for performing the operations of the present invention include general purpose digital computers or other similar digital devices. In all cases there should be borne in mind the distinction between the method operations in operating a computer and the method of computation itself. The present invention relates to method steps for operating a computer in processing electrical or other (e.g., mechanical, chemical) physical signals to generate other desired physical signals.

The present invention also relates to apparatus for performing these operations. This apparatus may be specially constructed for the required purposes or it may comprise a

general purpose computer as selectively activated or reconfigured by a computer program stored in the computer. The algorithms presented herein are not inherently related to a particular computer or other apparatus. In particular, various general purpose machines may be used with programs written in accordance with the teachings herein, or it may prove more convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description given below.

DETAILED DESCRIPTION OF THE INVENTION

The present invention discloses methods and apparatus for data storage, manipulation and retrieval. Although the present invention is described with reference to specific block diagrams, and table entries, etc., it will be appreciated by one of ordinary skill in the art that such details are disclosed simply to provide a more thorough understanding of the present invention. It will therefore be apparent to one skilled in the art that the present invention may be practiced without these specific details.

System Hardware

Referring to FIG. **1**, the hardware configuration of the present invention is conceptually illustrated. FIG. **1** illustrates an information storage and retrieval system structured in accordance with the teachings of the present invention. As illustrated, the information storage and retrieval system includes a computer **23** which comprises four major components. The first of these is an input/output (I/O) circuit **22**, which is used to communicate information in appropriately structured form to and from other portions of the computer **23**. In addition, computer **20** includes a central processing unit (CPU) **24** coupled to the I/O circuit **22** and to a memory **26**. These elements are those typically found in most computers and, in fact, computer **23** is intended to be representative of a broad category of data processing devices.

Also shown in FIG. **1** is a keyboard **30** for inputting data and commands into computer **23** through the I/O circuit **22**, as is well known. Similarly, a CD ROM **34** is coupled to the I/O circuit **22** for providing additional programming capacity to the system illustrated in FIG. **1**. It will be appreciated that additional devices may be coupled to the computer **20** for storing data, such as magnetic tape drives, buffer memory devices, and the like. A device control **36** is coupled to both the memory **26** and the I/O circuit **22**, to permit the computer **23** to communicate with multi-media system resources. The device control **36** controls operation of the multi-media resources to interface the multi-media resources to the computer **23**.

A display monitor **43** is coupled to the computer **20** through the I/O circuit **22**. A cursor control device **45** includes switches **47** and **49** for signally the CPU **24** in accordance with the teachings of the present invention. A cursor control device **45** (commonly referred to a "mouse") permits a user to select various command modes, modify graphic data, and input other data utilizing switches **47** and **49**. More particularly, the cursor control device **45** permits a user to selectively position a cursor **39** at any desired location on a display screen **37** of the display **43**. It will be appreciated that the cursor control device **45** and the keyboard **30** are examples of a variety of input devices which may be utilized in accordance with the teachings of the present invention. Other input devices, including for example, trackballs, touch screens, data gloves or other virtual reality devices may also be used in conjunction with the invention as disclosed herein.

System Architecture

FIG. **2** is a block diagram of the information storage and retrieval system of the present invention. As illustrated in the Figure, the present invention includes an internal database **52** that further includes a record oriented database **74** and a free-text database **76**. The database **52** may receive data from a plurality of external sources **50**, including word processing documents **58**, spreadsheets **60** and database files **62**. As will be described more fully below, the present invention includes an application support system that interfaces the external sources **50** with the database **52**.

To efficiently retrieve information stored in the database **52**, a plurality of indexes **54** including a keyword index **78** and other types of indexes such as phonetic, special sorting for other languages, and market specific such as chemical, legal and medical, store sorted information provided by the database **52**. To organize the information in the indexes **54**, a knowledge system **56** links information existing in the indexes **54**.

The organization illustrated in FIG. **2** is for conceptual purposes and, in actuality, the database **52**, the indexes **54** and the knowledge system **56** are stored in the same table, as will be described more fully below. This Specification will first describe the structure and features of the database **52**. Next, the Specification will describe the index **54** and its implementation for searching the database **52**. The Specification will then describe the knowledge system **56** that further enhances the index **54** by providing synonyms and other elements. Finally, the Specification will describe an interface between the external application programs **50** and the database **52**, including a novel structured word processor and a novel password scheme.

FIG. **3** illustrates the storage and retrieval structure of the present invention. The storage and retrieval structure of the present invention comprises a table **100**. The structure of the table **100** is a logical structure and not necessarily a physical structure. Thus, the memories **26** and **32** configured according to the teachings of the present invention need not store the table **100** contiguously.

The table **100** further comprises a plurality of rows **110** and a plurality of columns **120**. A row corresponds to a record while a column corresponds to an attribute of a record and the defining characteristics of the column are stored in a row **108**. The intersection of a row and a column comprises a particular cell.

Each row is assigned a unique object identification number (OID) stored in column **120** and each column also is assigned a unique OID, indicated in brackets and stored in row **108**. For example, row **110** has an OID equal to 1100 while the column **122** has an OID equal to 101. As will be described more fully below, the OID's for both rows and columns may be used as pointers and a cell **134** may store an OID. The method for assigning the OID's will also be discussed below.

As illustrated in FIG. **3**, each row, corresponding to a record, may include information in each column. However, a row need not, and generally will not, have data stored in every column. For example, row **110** corresponds to a company as shown in a cell **130**. Since companies do not have titles, cell **132** is unused.

The type of information associated with a column is known as a 'domain'. Standard domains supported in most database systems include text, number, date, and Boolean.

The present invention includes other types of domains such as the OID domain that points to a row or column. The present invention further supports 'user-defined' domains, whereby all the behavior of the domain can be determined by a user or programmer. For example, a user may configure a domain to include writing to and reading from a storage medium and handling operations such as equality testing and comparisons.

According to the present invention, individual cells may be accessed according to their row and column OID's. Using the cell as the unit of storage improves many standard data management operations that previously required the entire object or record. Such operations include versioning, security, hierarchical storage management, appending to remote partitions, printing, and other operations.

Column Definitions

Each column has an associated column definition, which determines the properties of the column, such as the domain of the column, the name of the column, whether the column is required and other properties that may relate to a column. The table **100** supports columns that include unstructured, free text data.

The column definition is stored as a record in the table **100** of FIG. **3**. For example, the "Employed By" column **126** has a corresponding row **136**. The addition or rows that correspond to columns renders the table **100** self-referential. New columns may be easily appended to the table **100** by creating a new column definition record. The new column is then immediately available for use in existing records.

Dates

Dates can be specified numerically and textually. An example of a numerical date is "11/6/67" and an example of a textual date is "Nov. 6, 1967." Textual entries are converted to dates using standard algorithms and lookup tables. A date value can store both original text and the associated date to which the text is converted, which allows the date value to be displayed in the format in which it was originally entered.

Numbers

Numeric values are classified as either a whole number (Integer) or fractional number. In the preferred embodiment, Integers are stored as variable length structures, which can represent arbitrarily large numbers. All data structures and indexes use this format which ensures that there are no limits in the system.

Fractional numbers are represented by a <numerator/denominator> pair of variable length integers. As with dates, a numeric value can store both the original text ("4½ inches") and the associated number (4.5). This allows the numeric value to be redisplayed in the format in which it was originally entered.

Type Definitions

A record can be associated with a 'record type'. The record type can be used simply as a category, but also can be used to determine the behavior of records. For example, the record type might specify certain columns that are required by all records of that type and, as with columns, the type definitions are stored as records in the table **100**. In FIG. **3**, column **122** includes the type definition for each record. The column **122** stores pointers to rows defining a particular column type. For example, the row **136** is a "Field" type

column and contains a pointer in a cell **133** to a row **135** that defines "Field" type columns. The "Type Column" **122** of the row **135** points to a type called "Type," which is defined in a row **140**. "Type" has a type column that points to itself.

Record types, as defined by their corresponding rows, may constrain the values that a record of that type may contain. For example, the record type 'Person' may require that records of type 'Person' have a valid value in the 'Name' column, the 'Phone' column, and any other columns. The type of a record is an attribute of the record and thus may change at any time.

Creating a Unique OID

As previously described, the system must generate a unique OID when columns and rows are formed. FIG. **4** is a flow chart of the method for assigning OID's.

At block **200** of FIG. **4**, the CPU **24** running the database program stored in the memory **26** requests a timestamp from the operating system. At block **210**, the system determines whether the received timestamp is identical to a previous timestamp. If the timestamps are identical, block **210** branches to block **220** and a tiebreaker is incremented to resolve the conflict between the identical timestamps. At block **222**, the system determines whether the tiebreaker has reached its limit, and, if so, the system branches to block **200** to retrieve a new time stamp. Otherwise, the system branches to block **214** where the system requests a session identification which is unique to the user session.

In the preferred embodiment, the session identification is derived from the unique serial number of the application installed on the users machine. For certain OID's which are independent of any particular machine, the session identification may be used to determine the type of object. For example, dates are independent of any particular machine, and so an OID for a date may have a fixed session identification.

Returning to block **210**, if the timestamps are not identical, control passes to block **212** where the tiebreaker is set to zero and control then passes to block **214**. As previously described, at block **214**, the system requests a session identification which is unique to the user session. Control then passes to block **216** where the session identification, timestamp and tiebreaker are combined into a bit array, which becomes the OID. Since the OID is a variable length structure, any number of bits may be used, depending on the precision required, the resolution of the operating system clock, and the number of users. In the preferred embodiment, the OID is 64 bits long where the timestamp comprises the first 32 bits, the tiebreaker comprises the next 10 bits and the session identification comprises 22 bits.

The particular type of OID and its length is constant throughout a single database but may vary between databases. A flag indicating which type of OID to be used may be embedded in the header of each database.

OID Domains

OID domains are used to store OID's, which are pointers to other records. An efficient query can use these OID's to go directly to another record, rather than searching through columns.

If a user wishes to search a column to find a record or records with a certain item in the column, and does not know the OID of the item, the present invention includes a novel technique for determining an OID from the textual description. Conversion from text to an OID may also be necessary

when a user is entering information into a record. For example, in FIG. 3, the user may be entering information in the "Employed By" column **126**, and wish to specify the text "DEXIS" and have it converted to OID #1100. For this purpose, special columns are required that provide a definition for how the search and conversion is performed.

FIG. **6** is a flow chart for searching the table **100** configured according to the structure illustrated in FIG. **5**. At block **150**, a user enters text through the keyboard **30** or mouse **45** for a particular column that the user wishes to search. At block **152**, the system retrieves the search path for the column to be searched from the information stored in column **146** as illustrated in FIG. **5**. Continuing with the above example, a cell **146** in the row **136** contains the search path information for the "Employed By" column **126** of FIG. **3**. The search path information for the "Employed By" field indicates that the folders called "\contacts" and "\departments" should be searched for a company with the label "DEXIS."

Returning to FIG. **5**, the system searches the table **100** according to the retrieved search path information. For each folder specified in the search path, the routine searches for a record that has an entry in the label column **124** of FIG. **2** that is the same as the text being searched for, and is of the same class, as indicated in column **122** of FIG. **3**. Folders will be further described below.

At block **156**, the system determines whether it has found any items matching the user's search text. If no items have been found, at block **158**, the system prompts the user on the display screen **37** to create a new record. If the user wishes to create a new record, control passes to block **162** and the system creates a new record. At block **164**, the OID of the new record is returned. If the user does not wish to create a new record, a "NIL" string is returned, as shown at block **160**.

If the system has located at least one item, the system determines whether it has found more than one item, as illustrated in block **166**. If only one item has been located, its OID is returned at block **168**. If more than one item has been located, the system displays the list of items to the user at block **170** and the user selects a record from the list. At block **172**, the OID of the selected record is returned, which, in the above example, is #1100, the OID of the record for the company "DEXIS."

In alternate embodiments, various features may be added to the search mechanism as described with reference to FIG. **6**. For example, further restrictions may be added to the search; the search may be related by allowing prefix matching or fuzzy matching instead of strict matching; and the search may be widened by using the 'associative search' techniques described below.

Two Way Synchronized Links

Records may have interrelationships and it is often desirable to maintain consistency between interrelated records. For example, a record including data for a company may include information regard employees of that company, as illustrated in row **110** of FIG. **3**. Similarly, the employees that work for that company may have a record that indicates, by a pointer, their employer, as illustrated by row **138** of FIG. **3**. Thus, the employee column of a company should point to employees whose employer column points to that company. The present invention includes a synchronization technique to ensure that whenever interrelated records are added or removed, the interrelationships between the columns are properly updated.

The system synchronizes interrelated records by adding a "Synchronize With" column **144** to the table **100** as illustrated in FIG. **5**. Since the value in the columns defines the relatedness between records, the rows **136** and **139** corresponding to columns contain information within the "Synchronize With" column **144** that indicates which other columns are to be synchronized with the columns corresponding to rows **136** and **139**. With reference to FIG. **5**, the "Employed By" column **126** is synchronized with the "Employees" column by an OID pointer in the "Synchronize With" column **144** to the "Employees" column, represented by row **139**. Similarly, the "Employees" column is synchronized with the "Employed By" column **136** by a pointer in the "Synchronize With" column **144** to the "Employed by" column **134**, represented by row **136**. Thus, whenever an employee changes companies, such that the employee's "Employed By" column changes, the "Employee" column of the previous employer is updated to eliminate the pointer to the ex-employee and, correspondingly, the addition of the employee in the "Employed By" field of the new employer. Synchronization may need to occur whenever a column is changed, whether by addition or subtraction of a reference to another column, or when entire records are added or eliminated from the table **100**.

FIG. **7a** is a flow chart for synchronizing records when a user adds or deletes a record. At block **180**, the system makes a backup of the original list of references to other rows, which are simply the OID's of those other rows, so that it can later determine which OIDS have been added or removed. Only these changes need to be synchronized. At block **182**, the system generates a new list of references by adding or deleting the specified OID. At block **184**, the system determines whether the relevant column is synchronized with another column. If it is not, then the system branches to block **186** and the update is complete. If the column is synchronized with another column, the system determines whether it is already in a synchronization routine. If this were not done, the routine would get into an endless recursive loop. If the system is already in a synchronization routine, the system branches to **190** and the update is complete.

Otherwise, the system performs actual synchronization. At block **192**, the system finds an OID that has been added or subtracted from the column (**C1**) of the record (**R1**) being altered. The system retrieves the record (**R2**) corresponding to the added or subtracted OID at block **194**. The system determines the synchronization column (**C2**) of the column (**C1**) at block **196** and locates that field in the added or subtracted OID. For example, if an employer is fired from a job, and the employer's "Employed By" field changed accordingly, the system would look up the value of the "Synchronize With" column **144** for the "Employees" column which is contained in the cell **147** as illustrated in FIG. **5**. Since cell **147** points to the "Employed By" field, the system locates the "Employed By" field of the record for the fired employee. At block **198** of FIG. **7a**, the located cell, (**R2:C2**), is updated by adding or subtracting the OID. Continuing with the above example, the "Employed By" field of the employee would be changed to no longer point to the previous employer by simply removing the employer's OID from that field. The system branches back to block **192** to update any other OID additions or subtractions. If the system has processed all of the OID's, then the routine exits as illustrated at blocks **200** and **202**.

FIG. **7b** illustrates the results of column synchronization of the "Employed By" field and the "Employees" field. As shown, the pointers in the records of these two columns are consistent with one another.

Columns Within Columns

A column may contain within it a reference to another column in the same record. For example, a 'name' column may contain a reference to both a 'first name' and a 'last name' column. The value of the 'name' column can then be reconstructed from the values of the other two columns. FIGS. **8***a* and **8***b* illustrate two possible implementations for reconstructing a value from one or more columns within the same record.

FIG. **8***a* illustrates a table **210** that includes a "First Name" column **220**, a "Last Name" column **222** and a "Name" column **224**. A record **226** for "John Smith" has the first name "John" in the "First Name" column **220** and the last name "Smith" in the column **222**. The name field **224** returns the text "The name is John Smith" by referencing the fields in brackets, according to the format <fieldRef field= 'Column Name'> as shown in column **224**.

FIG. **8***b* employs a variant of the referencing scheme illustrated in FIG. **8***a*. FIG. **8***a* illustrates a table **230** that includes a "First Name" column **232**, a "Last Name" column **234** and a "Name" column **236**. A record **238** for "John Smith" has the first name "John" in the "First Name" column **232** and the last name "Smith" in the column **234**. The name field **236** returns the text "The name is John Smith" by referencing the fields by defined variables 'fn' and 'ln' as shown in column **236**. The variables are defined according to the format variable:=fieldAt (parameter, 'Column Name') and the variables may be referenced in a return statement as shown in column **236**.

Record Contents

As previously described, a given row may contain values for any column. However, to determine all of the columns that might be used by a record would involve scanning every possible column. To avoid this problem, in the preferred embodiment, the table **100** illustrated in FIG. **3** includes a "RecordContents" column that indicates those columns within which a particular record has stored values.

FIG. **9** illustrates the table **100** with a "RecordContents" column **127** that includes pointers to the columns containing values for a particular record. For example, the "RecordContents" column **127** for row **110** has pointers to the column **124** and a column **125** but does not have a pointer to the column **126** because the row **110** does not have a value for the column **126**. As previously described, since every column has a corresponding row that defines the column, the "RecordContents" column **127** has a defining row **129**. Like any cell, the cell containing the record contents can be versioned, providing the ability to do record versioning.

Folders

To provide increased efficiency in managing information, the table **100** includes a data type defined as a folder. FIG. **10** illustrates the structure of a folder. As illustrated in the Figure, the table **100** includes a "Parent Folder" column **240** and a "Folder Children" column **242**. A folder has a corresponding record. For example, a folder entitled "Contacts" has a corresponding row **244** as illustrated in FIG. **10**. The "Folder Children" column **242** of the "Contacts" folder includes pointers to those records that belong to the folder. Similarly, those records that belong to a folder include a pointer to that folder in the "Parent Folder" column **240**.

The folder structure illustrated in FIG. **10** facilitates searching. As previously described, a column may be searched according to a folder specified in the column definition. If a folder is searched, the system accesses the record corresponding to the folder and then searches all of the records pointed to by that folder.

Further, the synchronization feature described above may be used to generate the list of items in a folder. For example, in FIG. **10**, the 'Folder Parent' and 'Folder Children' columns may be synchronized. When the 'Folder Parent' field **240** for record **138** is set to reference the 'Contacts' folder represented by row **244**, the list of items in the 'Contacts' folder ('FolderChildren') is automatically updated to store a reciprocal reference to record represented by row **138** by including its OID, 1100, in the "Folder Children" column **242**.

Text Indexing System

The present invention includes an indexing system that provides for rapid searching of text included in any cell in the table **100**. Each key phrase is extracted from a cell and stored in a list format according to a predefined hierarchy. For example, the list may be alphabetized, providing for very rapid searching of a particular name.

FIG. **11** illustrates the extraction of text from the table **100** to a list **250**. The list **250** is shown separately from the table **100** for purposes of illustration but, in the preferred embodiment, the list **250** comprises part of the table **100**. The list **250** stores cell identification numbers for each word in the list where a cell identification number may be of the format <record OID, column OID >. For example, the word "Ventura" occurs in cells **252**, **254** and **256** that correspond to different rows and different columns. The word "Ventura" in the list **250** contains a pointer, or cell identification number, to cells **252**, **254** and **256**.

Similarly, each cell stores the references to the key phrases within it using 'anchors'. As illustrated in FIG. **12**, an anchor contains a location (such as the start and stop offset within the text), and an identification number. Both the text and the anchor are stored in the cell **252**. Other kinds of domains also support anchors. For example, graphical images support the notion of 'hot spots' where the anchor position is a point on the image.

As previously described, each key phrase is stored as a record in the database and the OID of the record equals the identification number described with reference to FIG. **12**. One column stores the name of the key phrase and another stores the list of cell identification numbers that include that phrase. Key phrases may have comments of their own, which may also be indexed.

The sorted list **250** as illustrated in FIG. **11** is stored as a Folder, as illustrated in FIG. **13**. A cell identification field **274** maintains the cells that include the term corresponding to that record. The "Parent Folder" column **240** for each of the terms on the list **250** indicates that the parent folder is an index with a title "Natural." The "Natural" folder has a row **276** that has pointers in the "Folder Children" column **242** to all of the terms in the list **250**.

The "Natural" folder corresponds to an index sorted by a specific type of algorithm. Computer programs generally sort using a standard collating sequence such as ASCII. The present invention provides an improvement over this type of sorting and the improved sorting technique corresponds to the "Natural" folder. Records in the "Natural" folder are sorted according to the following rules:

1) A key phrase may occur at more than one point in the list. In particular:
   1a) Key phrases may be permuted and stored under each permutation. For example: 'John Smith' can be

stored under 'John' and also under 'Smith'. Noise words such as 'a' and 'the' are ignored in the permutation.

1b) Key phrases which are numeric or date oriented may be stored under each possible location. For example: '1984' can be stored under the digit '1984' and also under 'One thousand, nine hundred. . .', and 'nineteen eighty four'.

2) Numbers are sorted naturally. For example, '20' comes after '3' and before '100'.

3) Prefixes in key phrases are ignored. For example, 'The Big Oak' is sorted under 'Big'.

4) Key phrases are stemmed, so that 'Computers' and 'Computing' map to the identical key phrase record.

The preferred embodiment of the routine for generating positions for entering the key phrases into the 'Natural' folder is as follows:

1) Capitalize the key phrase to avoid case sensitivity problems. For example: 'John Smith the 1st' becomes 'JOHN SMITH THE 1ST'.

2) Each word in the key phrases is stemmed using standard techniques. Eg "COMPUTERS" becomes "COMPUT".

3) Permute the key phrase. This results in a new set of multiple key phrases based on the original key phrase. For example 'JOHN SMITH THE 1ST' produces the set {'JOHN SMITH THE 1ST'; 'SMITH THE 1ST JOHN'; 'THE 1ST JOHN SMITH'; '1ST JOHN SMITH THE'}.

4) Noise prefixes are eliminated. In the example above, the third entry, 'THE 1ST JOHN SMITH', is eliminated. If no phrases are left after elimination, the original phrase is used. For example, an entry for 'TO BE OR NOT TO BE" would be preserved even if all noise words were eliminated.

5) For each result, numbers and dates are expanded to all possible text representations, and text representations are converted to numeric. For example: '1ST JOHN SMITH THE' generates the set: {'1ST JOHN SMITH THE'; 'FIRST JOHN SMITH THE'}

6) Finally, each modified key phrase is used to determine the position of a reference to the main key phrase record, and an entry is made in the folder accordingly. For example, '1ST JOHN SMITH THE' is stored between '1' and '2', while 'FIRST JOHN SMIT THE' is stored after 'FIR' and before 'FIS.'

FIG. **22***a* illustrates the results of a prior art sorting algorithm while FIG. **22***b* illustrates the results of a sorting alogrithim according to the present invention.

Extracting the Key Phrases

To generate a sorted list, the system must first extract the key phrases or words from the applicable cells. The combination of structured information and text allows various combinations of key phrase extraction to be used. In full text extraction, every word is indexed, which is typical for standard text retrieval systems. In column extraction, the whole contents of the column are indexed which corresponds to a standard database system. According to a third type of extraction, automatic analysis, the contents of the text are analyzed and key phrases are extracted based on matching phrases, semantic context, and other factors. Finally, in manual selection extraction, the user or application explicitly marks the key phrase for indexing.

Date Indexing System

The date indexing scheme is very similar to the text indexing scheme as previously described. Important dates are extracted from the text and added to an 'Important Date' list. Each important date is represented by a 'Important Date' record. The 'Important Date' records are stored in a 'Important Dates' folder, which is sorted by date.

The important dates are extracted from the text. The system may search for numeric dates, such as '4/5/94' or date-oriented text, such as "Tomorrow", "next Tuesday" or "Christmas". FIG. **23** illustrates the correspondence between cells of the table of FIG. **2** and a sorted date index.

Important Date records are assigned special predetermined OIDS since they always have the same identity in any system. Assigning predetermined OID's to dates allows Important Dates to be shared across systems. The predetermined OID is generated by using a special session identification number that signifies that the OID is an Important Date. In this case, the timestamp represents the value of the Important Date itself, not the time that it was created.

Associative Queries

As previously described, a sorted key word list is generated from the text in cells and list stored in a folder whose records point to the text cells. The associations between the list of records with text and the list of key phrases is two-way since the cells that include text point to the key words. FIG. **14** illustrates this two way correspondence. Each record can point to multiple key phrases, and each key phrase can point to multiple records.

FIG. **15** is a graphical representation of the two way association between records and the key word list. Each record in a plurality of records **298** through **300** may point to one or more important word entries **310** through **312**. Similarly, each important word entry may point to one or more records. A single level search involves starting at one node (on either side of the graph) and following the links to the other side. For example, a user may wish to find the records including the word "Shasta." First, the important word index would be accessed to find the word "Shasta" and the records pointed to by this word would then be retrieved. This search is indicated by the arrows **314** and **316** where word "Shasta" corresponds to cell **318**. Similarly, a user may wish to locate all of the important words included in a particular record, indicated by the arrows **320** and **322** in FIG. **15**.

The search can be extended by repeatedly following the links back and forth to the desired level. FIG. **16***a* illustrates this concept. As an example, the term "Shasta" may correspond to a dog with extraordinary intelligence such that in one record, "Shasta" is described as a dog and another record, 'Shasta' is described as a genius. If the user wishes to find the words associated with 'Shasta', the system locates "Shasta" in the "Important Words" folder which points to the records including the word "Shasta." In turn, the records pointed to contain pointers to the "Important Words" list for each indexed word in the record. Since "Shasta" appears with "dog" and "genius" in the records, these words are retrieved by the system.

This type of searching may be extended indefinitely. FIG. **16***b* illustrates an additional level of searching. Continuing with the above example, the word "genius" may occur in records referring to Dirac, and the word "dog" associated with "Checkers," such that the multilevel search illustrated in FIG. **16***b* results in a retrieval of "Dirac" and "Checkers" when provided with the word "Shasta."

A relevance ranking can be created based on weights associated with each link and type of key word, and the records can be displayed in order of descending relevance.

In the preferred embodiment, if two or more nodes are used as the starting point, the relevance is based on the distance from all nodes. In this way, only nodes which are near all the initial nodes will have a high relevance. Many other relevance rankings apart from distance may be used.

To refine the search, filters can be used to constrain the links that are followed. For example, the search may be filtered such that only the type "Person" is listed such that, in the above example, Shasta will be associated with Dirac but not Checkers.

Knowledge Base and Thesaurus

The present invention includes a knowledge base and thesaurus to further improve searching capabilities.

Each important word record (term) included within the thesaurus contains a pointer to a 'concept' record. Each concept record contains pointers to other concept records, and to the terms that are included within the bounds of that concept. FIG. **17** illustrates the structure of the thesaurus. The table **100** includes a "Parent Concept" column **352**, a "Concept Name" column **354**, a "Synonyms" column **356**, a "More Specific Terms" column **358**, a "More General Terms" column **360** and a "See Also" column **362**. A concept record **350** defines the concept "IBM" and the Synonyms column **356** points to records that are synonymous with IBM, a record **364** with a label field with the value "IBM" and a record **366** with a label field with the value "International Business Machines." The records **364** and **366** have pointers in the "parent concept" field that point to the parent concept record **350**.

The thesaurus structure illustrated in FIG. **17** provides for greater flexibility than exact synonyms. The "More Specific Terms" column **358** of the concept record **350** associated with "IBM" points to a concept record **368** associated with the IBM PC with an assigned weight of 100%, where the weight percentage reflects the similarity between the initial term "IBM" and the related term "IBM PC." Similarly, the "More General Terms" column **360** of the concept record **350** associated with "IBM" points to a concept record **372** associated with Computer Companies with an assigned weight of 60%. The "See also" column points to a record associated with the concept "Microsoft" with a weight of 70%, where the weight percentage reflects the similarity between the initial term "IBM" and the related term "IBM PC."

The Thesaurus illustrated in FIG. **17** enhances the searching mechanisms previously described with reference to FIGS. **14**–**16***b*. The system first locates the record associated with a key word and locates the parent concept record pointed to by the key word record. The system may then follow some or all of the pointers in the columns **356**, **358**, **360** and **352** and return of the OID's stored in the 'Concept Name' column **354**.

Since key phrases and concepts are stored as records in this system, any other columns may be used to extend the knowledge and information stored therein. In particular, through the use of OID's, the system can store any kind of relationship, including relationships other than thesaural relationships, between key phrases, concepts and other records.

Application Support

The database of the present invention has been described without reference to its interface with applications that may use the invention as their primary storage and retrieval system. As previously described with reference to FIG. **2**, the present database includes an interface to support applications programs. Components in the application support system include external document support, hypertext, document management and workflow, calendaring and scheduling, security and other features.

Further, the present invention includes various user interface components that allow have been developed to provide full access to the structure of the database of the present invention. In particular, a new kind of structured word processor will be presented. The Specification will describe each component of the application support system separately.

External Documents

The present invention supports indexing of external documents. The table **100** stores the filenames of documents, such as word processor documents, where the contents of the files are not directly stored in the database. The documents names may be stored in a column with a specialized "External Document" domain. The external documents may reside in the mass memory **32** or on a multi-source that interfaces with the system through device control **36**.

To index documents external to the table **100**, prior to processing, an external document is converted into a plain text format. Key phrases are then extracted as previously described. In particular, fields in the text can be determined and mapped to fields within the database. For example, a 'Memo' document may contain the text: 'To: John Smith. From: Mary Doe'. This text can be mapped to the fields called 'to' and 'from', and the values of these fields set accordingly. The analysis of the text in this way can be changed for different types of external documents such as memos, legal documents, spread sheets, computer source code and any other type of document. For each extracted key phrase, a start and stop point within the text is determined. A list of anchors of the format previously described, <start, stop, key phrase> is generated by the parser and stored within the table **100** under the external document domain.

Viewing External Documents

When a user views an external document on the display screen **37**, the stored anchors are overlaid on top of the document such that it appears that the external document has been marked with hypertext. When the user clicks the switches **45** or **47** of the mouse **50** on a section of the external document display, the corresponding anchor is determined from the various start and stop coordinates. The OID of the key phrase corresponding to the anchor is stored within the anchor, and can be used for the purposes of retrieving the key phrase record or initiating a query as previously described.

Dynamic Hypertext

The present invention supports Hypertext. Hypertext systems typically associate a region of text with a pointer to another record, as illustrated in FIG. **18**. This creates a 'hard-coded' link between the source and the target. When user clicks on the source region, the target record is loaded and displayed. If the target record is absent, the hypertext jump will fail, possibly with serious consequences.

The present system uses a new approach based on a dynamic association between records. In the preferred embodiment, each hypertext region is associated with a key phrase, not a normal record. When the user clicks the

switches **45** or **47** of the mouse **50** on the source region, all the records associated with the key phrase are retrieved and ranked using any of the associative search techniques previously described. As illustrated in FIG. **19**, the application can then display on the display screen **37** either the highest ranked item, or present all the retrieved items and allow the user to pick the one to access.

In certain applications, the user may want to access a single 'default' item. This item can be determined automatically, by picking the item at the top of the dynamically generated list, or manually, by letting the user pick the item explicitly and then preserving this choice in the anchor itself.

The Generic Word Processor

The database of the present invention includes a novel Structured Word Processor that may be used in conjunction with the table **100**.

The structured word processor of the present invention uses the "boxes aud glue" paradigm introduced by Donald Knuth in T_EX. According to this paradigm, a page of text is created by starting with individual characters and concateuating the characters to form larger units, called "boxes," and then combining these boxes into yet larger boxes. FIG. **20***a* illustrates three character boxes **400**, **402** and **404** concatenated to form a word box **406**. FIG. **20***b* illustrates four word boxes **410**, **412**, **414** and the word box **406** combined to form a horizontal line box **408**. Horizontal boxes are used for words and other text tokens that are spaced horizontally inside another box, such as a line (or column width). FIG. **20***c* illustrates the combination of the horizontal line box **408** with another horizontal line box **4242** to form a vertical box **420**. Vertical boxes are used for paragraphs and other objects that are spaced vertically inside other boxes, such as page height.

Boxes may be attached to other boxes with "glue." The glue can stretch or shrink, as needed. For example, in a justified sentence, the white space between words is stretched to force the words to line up at the right edge of the column. Glue can be used for between-character (horizontal) spacing, between-word (horizontal) spacing including "tab" glue, that "sticks" to tab markings. Glue may also be used for between-line (vertical) spacing and between-paragraph (vertical) spacing.

When a record of the table **100** is edited, each word and field definition is converted into boxes. The system organizes these boxes into a tree structure of line boxes and paragraph boxes, as illustrated in FIG. **21**. Shown there is a record hierarchy **460**, corresponding to the hierarchy of a record, and a layout hierarchy **470**, corresponding to the hierarchy of a layout such as a document generated according to the word processor described with reference to FIGS. **20***a*–**20***c*. The record structure hierarchy **460** represents the record structure of the table **100** where a record **462** corresponds to a row in the table **100** and the record **462** includes a plurality of attributes, including attribute **464**, that correspond to the columns of the table **100**. In turn, the attributes may include a variety of items. For example, the attribute **464** includes text, represented by block **466**, field references represented by block **468** and other items as shown.

The layout hierarchy **470** comprises a document **472** which in turn comprises a plurality of pages, including page **474**. The page **474** comprises a plurality of paragraphs including paragraphs **430** and **431** and the paragraph **430** comprises a plurality of lines, including lines **432** and **434**. The paragraph **431** includes line **436**.

The word processor of the present invention allows the document **472** to be inserted into the record **462** by providing a plurality of boxes, including boxes **438**, **440** and **442**, common to both the record structure hierarchy **460** and the layout hierarchy **470**. For example, the box **438** corresponds to part of the line **432** and comprises part of the text of attribute **464** as illustrated by block **466**. Similarly, the box **440** corresponds to part of the line **434** and may comprise a field reference as indicated by block **468**. Thus, the shared box structure as illustrated in FIG. **21** allows any type of word processing document to interface with any record in the table **100**.

Conceptually, each box is kept as a bitmap, and its height and width are known, so the system displays the tree structure **450** by displaying all of the bitmaps corresponding to the boxes in the tree. If the tree is changed, for example, by adding a new word, only the new word box and a relatively small number of adjacent boxes need be recalculated. Similarly, line breaks or restructuring of a paragraph does not alter most of the word boxes, which may be reused, and only the lineboxes need be recalculated.

To edit the tree structure **450** as illustrated in FIG. **21**, a user may click a cursor on a part of the text. The system locates the word box or glue that is being edited by a recursively descending through the tree structure **450**.

The word processor supports multiple fonts and special effects such as subscripts, dropcaps and other features including graphic objects. A word in a different font than a base font is in a different box and may have a different height from other boxes on a line. The height of a linebox the height of the largest wordbox within it. Effects within a word can be handled by breaking a word into subboxes with no glue between them. Again, the height of a wordbox is the height of the largest box within it. Graphic objects, such as bitmaps, may be treated and formatted as a fixed width box.

The word processor of the present invention may be used to edit records in the table **100**. The text associated with each field in a record can be considered a "paragraph" for the purposes of inter-field spacing, text flow within a field, and other formatting parameters. Storing all the fields in the same way during text-editing allows the movement of text and "flow" to appear natural.

As previously described, the text being edited is divided into fields, with each field corresponding to a column in the underlying database. Unlike a traditional static data entry form, the positions and sizes of the attributes are not fixed but are dynamic and all the features of a word-processor such as fonts and embedded graphics are available to edit the record fields.

Similarly, all of the features of a database such as lookups and mailmerge are available to the word processor. All of the attributes that apply to data entry for a particular field are enforced by the word processor. Such attributes might include a mask (such as ###-####), existence requirements, range and value constraints, etc. The fields can be explicitly labelled, or hidden and implied.

The word processor of the present invention allows existing fields to be added by typing the prefix of a field name and pressing a button. The system then completes the rest of the field name automatically.

The word processor of the present invention supports other database features. For example, new fields can be created by a user by using a popup dialog box. Similarly, references to other records or important words can be added by a dialog box. With particular regard to the table **100** of the present invention, OID references may support fields within

other fields and a particular field within other fields supports the use of 'templates,' where a template is a list of field references embedded in text. For example, the template "Enter the first name here <fieldref id=firstName> and the last name here <fieldref id=lastName>" would appear to the user as "Enter the first name here: John and the last name here: Doe." Templates allow a user to build dynamic forms quickly and easily without having to use complicated form drawing tools.

The user interface for the word processor of the present invention allows a user to switch between two modes of data entry. The word-processor of the present invention is used for flexible entry into one record at a time, while a columnar view is used for entering data in columns. The user can switch back and forth between these two views with no loss of data and switching from the word processor to the columnar view will cause the fields that were entered in the single item to become the columns to be displayed in the columnar view.

Finally, the 'fields within fields' that are apparent in the word processor view become separated into columns in a columnar view. The user can then make changes in columnar mode, and then, when switching back to the word processor view, the columns become combined once again.

Passwords

It is often required that access to particular data items be restricted to certain users. In order to apply these restrictions, an information management system must determine the identity of the user requesting access. This is currently done in two ways, physically measuring a unique quality of the uses of requesting information from the user, most current information management systems rely on the second approach, by using 'passwords'. However, to avoid security problems with a password system, three guidelines are applied to passwords:

a) the password should not be made of common words, because an aggressor can use a brute force approach and a dictionary to guess the password;

b) the password should be longer rather than shorter; and

c) the password should be changed often, so that even if is stolen it will not be valid for long.

Finally, a password should never be written down or embedded into a login script and should always be interactive.

According to the present password system, a user's identity is determined through an extensive question and answer session. The responses to certain personal questions very quickly identify the user with high accuracy. Even an accurate mimic will eventually fail to answer correctly if the question and answer session is prolonged.

For example, sample questions might be: 'What is your favorite breakfast cereal?'; 'Where were you in April 1990?' 'What color is your toothbrush?'. These questions are wide ranging and hard to mimic. Furthermore, the correct responses are natural English sentences, with an extremely large solution space, so that a brute force approach is unlikely to be successful.

To improve the effectiveness of the response, an exact matching of user response and stored answer is not required and 'fuzzy' and 'associative' matching can be used according to the synonym, thesaurus and other features of the present invention.

According to the password system of the present invention, the user creates the list of questions and corresponding answers, which are then stored. Because the user has complete control over the questions, the user may find

the process of creating the questions and answers enjoyable, and as a result, change the questions and answer list more frequently, further enhancing system security.

According to the preferred embodiment, a user creates a list of 50–100 questions and answers that are encrypted and stored. The questions can be entirely new, or can be based on a large database of interesting questions. When the user logs on the system, the system randomly selects one of the questions related to that user and presents the question to the user. The user then types in a response, which is matched against the correct answer. The matching can be fuzzy and associative, as described above. If the response matches correctly, access is allowed.

In an alternate embodiment, more security may be provided by repeatedly asking questions until a certain risk threshold is reached. For example, if the answer to 'What color is your toothbrush?' is the single word 'Red', then brute force guessing may be effective in this one case. In this scenario, repeatedly asking questions will diminish the probability of brute force success.

Summary

While the invention has been described in conjunction with the preferred embodiment, it is evident that numerous alternatives, modifications, variations and uses will be apparent to those skilled in the art in light of the foregoing description. Many other adaptations of the present invention are possible.

What is claimed is:

1. A data storage and retrieval system for a computer having a memory, a central processing unit and a display, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein at least one of said records has an OID equal to the OID of a corresponding one of said attribute sets, and at least one of said records includes attribute set information defining each of said attribute sets.

2. The system of claim 1 wherein said attribute set information defines one of said attribute sets to contain information for enabling determination of OIDs from text entry.

3. The system of claim 1 wherein said one of said attribute sets contains information including a search path that references a folder, said folder including a group of records of a similar type.

4. The system of claim 1, wherein said attribute set information defines one of said attribute sets to contain information for synchronizing two attribute sets reciprocally.

5. The system of claim 4 wherein said one of said attribute sets contains information including reciprocal pointers to said two attribute sets.

6. The system of claim 1 wherein:

at least one of said plurality of records includes information defining the type of a different record; and

at least one of said plurality of records includes a cell that contains a pointer to said record including record type information.

7. The system of claim **1** wherein at least one of said attribute sets defines cells that include a plurality of pointers to other attribute sets within the same record, said pointers indicating those attribute sets within the same record that contain defined values.

8. The system of claim **1** wherein at least one of said records is a folder type record, said folder type record including at least one cell that contains data and a plurality of pointers to a plurality of other records included within said folder.

9. The system of claim **8** wherein said plurality of other records included within said folder each includes a cell that contains a pointer to said folder type record.

10. The system of claim **1** wherein said OID's are variable length and include data related to a session identification number and a timestamp.

11. A data storage and retrieval system for a computer having a memory, a central processing unit and a display, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, at least one of said records contains a cell having a pointer to a different record and at least one of said records includes attribute set information defining each of said attribute sets; and

means for searching said table for said pointer.

12. The system of claim **11** wherein at least one of said attribute sets defines cells that include a plurality of pointers to other attribute sets within the same record, said pointers indicating those attribute sets within the same record that contain defined values.

13. The system of claim **11** wherein at least one of said records is a folder type record, said folder type record including at least one cell that contains data and a plurality of pointers to a plurality of other records included within said folder.

14. The system of claim **13** wherein said plurality of other records included within said folder each includes a cell that contains a pointer to said folder type record.

15. A data storage and retrieval system for a computer having a memory, a central processing unit and a display, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, at least one of said plurality of records contains a cell having a pointer to a different record and at least one of said plurality of records includes information defining the type of a different record; and

means for searching said table for said pointer.

16. A data storage and retrieval system for a computer having a memory, a central processing unit and a display, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein said OID's are variable length.

17. A data storage and retrieval system for a computer having a memory, a central processing unit and a display, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record; and

means for indexing data stored in said table.

18. The system of claim **17** wherein said indexing means further comprises:

means for searching a plurality of cells within said table for a key word, said searching means capable of searching a attribute set containing unstructured text and a attribute set containing structured data; and

means for inserting into said table a record corresponding to said key word.

19. The system of claim **18** wherein:

said inserted record includes a cell that contains a pointer to a searched cell that contains the keyword corresponding to said inserted record; and

said searched cell that contains a keyword corresponding to said inserted record contains a pointer to said inserted record.

20. The system of claim **19** wherein said pointer to said searched cell includes the OID's of the attribute set and record defining said searched cell.

21. The system of claim **19** wherein said searched cell includes an anchor that marks said key word.

22. The system of claim **18** wherein one of said plurality of records of said table includes a folder type record that includes at least one pointer to said key word.

23. The system of claim **18** wherein said searching means further includes:

means for searching for every word in a text cell;

means for searching for every entry in a attribute set;

means for searching for data based on automatic analysis; and

means for searching for data marked by a user.

24. A data storage and retrieval system for a computer having a memory, a central processing unit and a display, comprising:

means for configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment, at least one of said cells includes a pointer to an index record;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record; and

means for indexing data stored in said table.

25. The system of claim **24** wherein said indexing means further comprises:

means for searching said table for a key word; and

means for creating an index record for said key word, said index record including one or more pointers to a cell in said table that contains said key word.

26. The system of claim **25** further including querying means, said querying means further including:

index look-up means for locating said index record according to the query of a user; and

record retrieval means for retrieving at least one cell in said table pointed to by said located index record.

27. The system of claim **26** wherein said index look-up means includes means for locating said index record pointed to by said at least one retrieved cell.

28. The system of claim **27** wherein said index look-up means and said record retrieval means each includes weighing means for weighing key words and retrieved cells according to pre-defined search criteria.

29. The system of claim **27** wherein said index look-up means and said record retrieval means each includes filtering means for filtering key words and retrieved cells according to pre-defined search criteria.

30. The system of claim **25** wherein said indexing means further includes means for indexing external documents.

31. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein at least one of said records has an OID equal to the OID of a corresponding one of said attribute sets, and at least one of said records includes attribute set information defining each of said attribute sets.

32. The method of claim **31** wherein said attribute set information defines one of said attribute sets to contain information for enabling determination of OIDs from text entry.

33. The method of claim **31** wherein said one of said attribute sets contains information including a search path that references a folder, said folder including a group of records of a similar type.

34. The method of claim **31** wherein said attribute set information defines one of said attribute sets to contain information for synchronizing two attribute sets reciprocally.

35. The method of claim **34** wherein said one of said attribute sets contains information including reciprocal pointers to said two attribute sets.

36. The method of claim **31** wherein:

at least one of said plurality of records includes information defining the type of a different record; and

at least one of said plurality of records includes a cell that contains a pointer to said record including record type information.

37. The method of claim **31** wherein at least one of said attribute sets defines cells that include a plurality of pointers to other attribute sets within the same record, said pointers indicating those attribute sets within the same record that contain defined values.

38. The method of claim **37** wherein at least one of said records is a folder type record, said folder type record including at least one cell that contains data and a plurality of pointers to a plurality of other records included within said folder.

39. The method of claim **38** wherein said plurality of other records included within said folder each includes a cell that contains a pointer to said folder type record.

40. The method of claim **31** wherein said OID's are variable length and include data related to a session identification number and a timestamp.

41. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein at least one of said records has an OID equal to the OID of a corresponding one of said attribute sets, and at least one of said records includes attribute set information defining each of said attribute sets; and

searching said table for said pointer.

42. The method of claim **41** wherein at least one of said attribute sets defines cells that include a plurality of pointers to other attribute sets within the same record, said pointers indicating those attribute sets within the same record that contain defined values.

43. The method of claim **41** wherein at least one of said records is a folder type record, said folder type record including at least one cell that contains data and a plurality of pointers to a plurality of other records included within said folder.

44. The method of claim **43** wherein said plurality of other records included within said folder each includes a cell that contains a pointer to said folder type record.

45. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein at least one of said records contains a cell that contains a pointer to a different record and at least one of said plurality of records includes information defining the type of a different record; and

searching said table for said pointer.

46. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set; and

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record, wherein said OID's are variable length.

47. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;

a plurality of records, each said record inclnding a series of cells having the same first address segment, each said record including an OID to identify each said record; and

indexing data stored in said table.

48. The method of claim **47** wherein the step of indexing data stored in said table further comprises:

searching a plurality of cells within said table for a key word, said cells containing unstructured text or structured data; and

inserting a record into said table corresponding to said key words.

49. The method of claim **48** wherein:

said inserted record includes a cell that contains a pointer to a searched cell that contains the keyword corresponding to said inserted record; and

said searched cell that contains a keyword corresponding to said inserted record contains a pointer to said inserted record.

50. The method of claim **49** wherein said pointer to said searched cell includes the OID's of the attribute set and record defining said searched cell.

51. The method of claim **49** wherein said searched cell includes an anchor that marks said key word.

52. The method of claim **48** wherein one of said plurality of records of said table includes a folder type record that includes at least one pointer to said key word.

53. The method of claim **48** wherein said step of searching a plurality of cells within said table for a key word further comprises the steps of:

searching for every word in a text cell;

searching for every entry in a attribute set;

searching for data based on automatic analysis; and

searching for data marked by a user.

54. A method for storing and retrieving data in a computer system having a memory, a central processing unit and a display, comprising the steps of:

configuring said memory according to a logical table, said logical table including:

a plurality of cells, each said cell having a first address segment and a second address segment, at least one of said cells includes a pointer to an index record;

a plurality of attribute sets, each said attribute set including a series of cells having the same second address segment, each said attribute set including an object identification number (OID) to identify each said attribute set;

a plurality of records, each said record including a series of cells having the same first address segment, each said record including an OID to identify each said record; and

indexing data stored in said table.

55. The method of claim **54** wherein said step of indexing data further comprises the steps of:

searching said table for a key word; and

creating an index record for said key word, said index record including one or more pointers to a cell in said table that contains said key word.

56. The method of claim **55** further comprising the steps of:

locating said index record according to the query of a user; and

retrieving at least one cell in said table pointed to by said located index record.

57. The method of claim **56** wherein said step of locating said index record further comprises the step of:

locating said index record pointed to by said at least one retrieved cell.

58. The method of claim **57** wherein said step of locating said index record further comprises the step of:

weighing key words and retrieved cells according to pre-defined search criteria.

59. The method of claim **57** wherein said step of locating said index record further comprises the step of:

filtering key words and retrieved cells according to pre-defined search criteria.

60. The method of claim **54** wherein said step of indexing data further comprises the step of:

indexing external documents.

* * * * *

CERTIFICATE OF COMPLIANCE

1. This brief complies with the type-volume limitation of Federal Rule of Appellate Procedure 32(a)(7)(B). The brief contains 13,972 words, excluding the portions exempted by Federal Rule of Appellate Procedure 32(a)(7)(B)(iii).

2. This brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5) and the type style requirements of Federal Rule of Appellate Procedure 32(a)(6). The brief has been prepared in a proportionally spaced typeface using Microsoft® Word 2010 and 14-point Times New Roman type.

Dated: November 20, 2015.

/s/Chad S. Campbell
Chad S. Campbell

CERTIFICATE OF AUTHORITY AND PROOF OF SERVICE

I certify that I have the authority of my co-counsel Chad S. Campbell to file this brief with his electronic signature.

In accordance with Federal Rule of Appellate Procedure 25 and Federal Circuit Rule 25, I further certify that I caused this brief to be served via the Federal Circuit's CM/ECF system on counsel of record for all parties.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct.

Dated: November 20, 2015.

/s/Dan L. Bagatell
Dan L. Bagatell

’604 Patent Claims

32. The method of claim 31 wherein said logical column information defines one of said logical columns to contain information for enabling determination of OIDs from text entry.

36. The method of claim 31 wherein:

at least one of said plurality of logical rows includes information defining the type of a different logical row; and

at least one of said plurality of logical rows includes a logical cell that contains a pointer to said logical row including logical row type information.

37. The method of claim 31 wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, and pointers indicating those logical columns within the same record that contain defined values.

43. The method of claim 42 wherein at least one of said logical columns defines logical cells that include a plurality of pointers to other logical columns within the same record, said pointers indicating those logical columns within the same record that contain defined values.

55. The method of claim 54 wherein said step of indexing data further comprises the steps of:

searching said table for a key word; and

creating an index record for said key word, said index record having one or more pointers to a logical cell in said table that contains said key word.

56. The method of claim 55 further comprising the steps of:

locating said index record according to the query of a user;

retrieving at least one logical cell in said table pointed to by said located index record.

60. The method of claim 54 wherein said step of indexing data further comprises the step of:

indexing external documents.